Graph Neural Networks for Natural Language Processing: A Survey

Suggested Citation: Lingfei Wu[4], Yu Chen[1], Kai Shen[5], Xiaojie Guo, Hanning Gao, Shucheng Li[6], Jian Pei and Bo Long (2018), "Graph Neural Networks for Natural Language Processing:
A Survey", : Vol. xx, No. xx, pp 1–18. DOI: 10.1561/XXXXXXXXX.

Lingfei Wu
JD.COM Silicon Valley Research Center, USA
lwu@email.wm.edu

Yu Chen
Rensselaer Polytechnic Institute, USA
hugochan2013@gmail.com

Kai Shen
Zhejiang University, China
shenkai@zju.edu.cn

Xiaojie Guo
JD.COM Silicon Valley Research Center, USA
xguo7@gmu.edu

Hanning Gao
Central China Normal University, China
ghnqwerty@gmail.com

Shucheng Li
Nanjing University, China
shuchengli@smail.nju.edu.cn

Jian Pei
Simon Fraser University, Canada
jpei@cs.sfu.ca

Bo Long
JD.COM, China
bo.long@jd.com

Boston — Delft

Contents

0.1 Introduction 2
0.2 Graph Based Algorithms for NLP 5
0.3 Graph Neural Networks 10
0.4 Graph Construction Methods for NLP 19
0.5 Graph Representation Learning for NLP 39
0.6 GNN Based Encoder-Decoder Models 61
0.7 Applications 78
0.8 General Challenges and Future Directions 126
0.9 Conclusions 132

References **133**

Graph Neural Networks for Natural Language Processing: A Survey

Lingfei Wu*[1], Yu Chen*[2], Kai Shen**[3], Xiaojie Guo[4], Hanning Gao[5], Shucheng Li†[6], Jian Pei[7] and Bo Long[8]

[1]*JD.COM Silicon Valley Research Center, USA; lwu@email.wm.edu*
[2]*Rensselaer Polytechnic Institute, USA; hugochan2013@gmail.com*
[3]*Zhejiang University, China; shenkai@zju.edu.cn*
[4]*JD.COM Silicon Valley Research Center, USA; xguo7@gmu.edu*
[5]*Central China Normal University, China; ghnqwerty@gmail.com*
[6]*Nanjing University, China; shuchengli@smail.nju.edu.cn*
[7]*Simon Fraser University, Canada; jpei@cs.sfu.ca*
[8]*JD.COM, China; bo.long@jd.com*

ABSTRACT

Deep learning has become the dominant approach in addressing various tasks in Natural Language Processing (NLP). Although text inputs are typically represented as a sequence of tokens, there is a rich variety of NLP problems that can be best expressed with a graph structure. As a result, there is a surge of interest in developing new deep learning techniques on graphs for a large number of NLP tasks. In this survey, we present a comprehensive overview on *Graph Neural Networks (GNNs) for Natural Language Processing*. We propose a new taxonomy of GNNs for NLP,

*Both authors contributed equally to this research.
**This research is done when Kai Shen is an intern at JD.COM.
†Shucheng Li is also with National Key Lab for Novel Software Technology, Nanjing University.

Lingfei Wu[1], Yu Chen[1], Kai Shen[2], Xiaojie Guo, Hanning Gao, Shucheng Li[3], Jian Pei and Bo Long (2018), "Graph Neural Networks for Natural Language Processing: A Survey", : Vol. xx, No. xx, pp 1–18. DOI: 10.1561/XXXXXXXXX.

which systematically organizes existing research of GNNs for NLP along three axes: graph construction, graph representation learning, and graph based encoder-decoder models. We further introduce a large number of NLP applications that exploits the power of GNNs and summarize the corresponding benchmark datasets, evaluation metrics, and open-source codes. Finally, we discuss various outstanding challenges for making the full use of GNNs for NLP as well as future research directions. To the best of our knowledge, this is the first comprehensive overview of Graph Neural Networks for Natural Language Processing.

0.1 Introduction

Deep learning has become the dominant approach in coping with various tasks in Natural Language Processing (NLP) today, especially when operated on large-scale text corpora. Conventionally, text sequences are considered as a bag of tokens such as BoW and TF-IDF in NLP tasks. With recent success of Word Embeddings techniques (Mikolov *et al.*, 2013; Pennington *et al.*, 2014), sentences are typically represented as a sequence of tokens in NLP tasks. Hence, popular deep learning techniques such as recurrent neural networks (Schuster and Paliwal, 1997) and convolutional neural networks (Krizhevsky *et al.*, 2012) have been widely applied for modeling text sequence.

However, there is a rich variety of NLP problems that can be best expressed with a graph structure. For instance, the sentence structural information in text sequence (i.e. syntactic parsing trees like dependency and constituency parsing trees) can be exploited to augment original sequence data by incorporating the task-specific knowledge. Similarly, the semantic information in sequence data (i.e. semantic parsing graphs like Abstract Meaning Representation graphs and Information Extraction graphs) can be leveraged to enhance original sequence data as well. Therefore, these graph-structured data can encode complicated pairwise relationships between entity tokens for learning more informative representations.

Unfortunately, deep learning techniques that were disruptive for Euclidean data (e.g, images) or sequence data (e.g, text) are not immediately applicable to graph-structured data, due to the complexity of graph data such as irregular

structure and varying size of node neighbors. As a result, this gap has driven a tide in research for deep learning on graphs, especially in development of graph neural networks (GNNs) (Wu *et al.*, 2022; Kipf and Welling, 2016; Defferrard *et al.*, 2016; Hamilton *et al.*, 2017a).

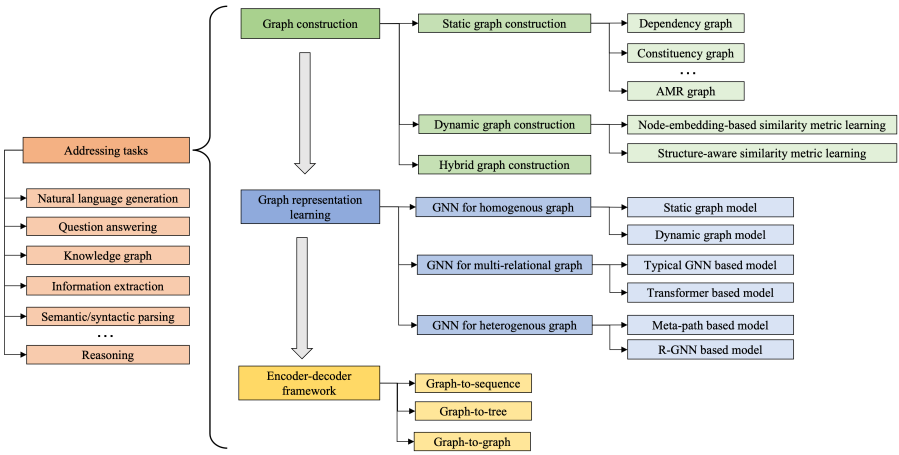


Figure 1: The taxonomy, which systematically organizes GNNs for NLP along four axes: graph construction, graph representation learning, encoder-decoder models, and the applications.

This wave of research at the intersection of deep learning on graphs and NLP has influenced a variety of NLP tasks (Liu and Wu, 2022). There has seen a surge of interests in applying and developing different GNNs variants and achieved considerable success in many NLP tasks, ranging from classification tasks like sentence classification (Henaff *et al.*, 2015; Huang and Carley, 2019), semantic role labeling (Luo and Zhao, 2020; Gui *et al.*, 2019), and relation extraction (Qu *et al.*, 2020; Sahu *et al.*, 2019), to generation tasks like machine translation (Bastings *et al.*, 2017; Beck *et al.*, 2018a), question generation (Pan *et al.*, 2020; Sachan *et al.*, 2020), and summarization (Fernandes *et al.*, 2019; Yasunaga *et al.*, 2017). Despite the successes these existing research has achieved, deep learning on graphs for NLP still encounters many challenges, namely,

- Automatically transforming original text sequence data into highly graph-structured data. Such challenge is profound in NLP since most of the NLP tasks involving using the text sequences as the original inputs. Automatic graph construction from the text sequence to utilize

the underlying structural information is a crucial step in utilizing graph neural networks for NLP problems.

- Properly determining graph representation learning techniques. It is critical to come up with specially-designed GNNs to learn the unique characteristics of different graph-structures data such as undirected, directed, multi-relational and heterogeneous graphs.
- Effectively modeling complex data. Such challenge is important since many NLP tasks involve learning the mapping between the graph-based inputs and other highly structured output data such as sequences, trees, as well as graph data with multi-types in both nodes and edges.

In this survey, we will present for the first time a comprehensive overview of *Graph Neural Networks for Natural Language Processing*. Our survey is timely for both Machine Learning and NLP communities, which covers relevant and interesting topics, including automatic graph construction for NLP, graph representation learning for NLP, various advanced GNNs-based encoder-decoder models (i.e. graph2seq, graph2tree, and graph2graph) for NLP, and the applications of GNNs in various NLP tasks. We highlight our main contributions as follows:

- We propose a new taxonomy of GNNs for NLP, which systematically organizes existing research of GNNs for NLP along four axes: graph construction, graph representation learning, and graph based encoder-decoder models.
- We present the most comprehensive overview of the state-of-the-art GNNs-based approaches for various NLP tasks. We provide detailed descriptions and necessary comparisons on various graph construction approaches based on the domain knowledge and semantic space, graph representation learning approaches for various categories of graph-structures data, GNNs-based encoder-decoder models given different combinations of inputs and output data types.
- We introduce a large number of NLP applications that are exploiting the power of GNNs, including how they handle these NLP tasks along three key components (i.e., graph construction, graph representation learning, and embedding initialization), as well as providing corresponding benchmark datasets, evaluation metrics, and open-source codes.

- We outline various outstanding challenges for making the full use of GNNs for NLP and provides discussions and suggestions for fruitful and unexplored research directions.

The rest of the survey is structured as follows. Section 0.2 reviews the NLP problems from a graph perspective, and then briefly introduces some representative traditional graph-based methods for solving NLP problems. Section 0.3 elaborates basic foundations and methodologies for graph neural networks, which are a class of modern neural networks that directly operate on graph-structured data. We also provide a list of notations used throughout this survey. Section 0.4 focuses on introducing two major graph construction approaches, namely static graph construction and dynamic graph construction for constructing graph structured inputs in various NLP tasks. Section 0.5 discusses various graph representation learning techniques that are directly operated on the constructed graphs for various NLP tasks. Section 0.6 first introduces the typical Seq2Seq models, and then discusses two typical graph-based encoder-decoder models for NLP tasks (i.e., graph-to-tree and graph-to-graph models). Section 0.7 discusses 12 typical NLP applications using GNNs bu providing the summary of all the applications with their sub-tasks, evaluation metrics and open-source codes. Section 0.8 discusses various general challenges of GNNs for NLP and pinpoints the future research directions. Finally, Section 0.9 summarizes the paper. The taxonomy, which systematically organizes GNN for NLP approaches along four axes: graph construction, graph representation learning, encoder-decoder models, and the applications is illustrated in Fig.1.

0.2 Graph Based Algorithms for NLP

In this section, we will first review the NLP problems from a graph perspective, and then briefly introduce some representative traditional graph-based methods for solving NLP problems.

0.2.1 Natural Language Processing: A Graph Perspective

The way we represent natural language reflects our particular perspective on it, and has a fundamental influence on the way we process and understand it. In general, there are three different ways of representing natural language.

The most simplified way is to represent natural language as a bag of tokens. This view of natural language completely ignores the specific positions of tokens appearing in text, and only considers how many times a unique token appears in text. If one randomly shuffles a given text, the meaning of the text does not change at all from this perspective. The most representative NLP technique which takes this view is topic modeling (Blei *et al.*, 2003) which aims to model each input text as a mixture of topics where each topic can be further modeled as a mixture of words.

A more natural way is to represent natural language as a sequence of tokens. This is how human beings normally speak and write natural language. Compared to the above bag perspective, this view of natural language is able to capture richer information of text, e.g., which two tokens are consecutive and how many times a word pair co-occurs in local context. The most representative NLP techniques which take this view include the linear-chain CRF (Lafferty *et al.*, 2001) which implements sequential dependencies in the predictions, and the word2vec (Mikolov *et al.*, 2013) which learns word embeddings by predicting the context words of a target word.

The third way is to represent natural language as a graph. Graphs are ubiquitous in NLP. While it is probably most apparent to regard text as sequential data, in the NLP community, there is a long history of representing text as various kinds of graphs. Common graph representations of text or world knowledge include dependency graphs, constituency graphs, AMR graphs, IE graphs, lexical networks, and knowledge graphs. Besides, one can also construct a text graph containing multiple hierarchies of elements such as document, passage, sentence and word. In comparison with the above two perspectives, this view of natural language is able to capture richer relationships among text elements. As we will introduce in next section, many traditional graph-based methods (e.g., random walk, label propagation) have been successfully applied to challenging NLP problems including word-sense disambiguation, name disambiguation, co-reference resolution, sentiment analysis, and text clustering.

0.2.2 Graph Based Methods for Natural Language Processing

In this previous subsection, we have discussed that many NLP problems can be naturally translated into graph-based problems. In this subsection, we will

introduce various classical graph-based algorithms that have been successfully applied to NLP applications. Specifically, we will first briefly illustrate some representative graph-based algorithms and their applications in the NLP field. And then we further discuss their connections to GNNs. For a comprehensive coverage of traditional graph-based algorithms for NLP, we refer the readers to (Mihalcea and Radev, 2011).

Random Walk Algorithms

Approach Random walk is a class of graph-based algorithms that produce random paths in a graph. In order to do a random walk, one can start at any node in a graph, and repeatedly choose to visit a random neighboring node at each time based on certain transition probabilities. All the visited nodes in a random walk then form a random path. After a random walk converges, one can obtain a *stationary distribution* over all the nodes in a graph, which can be used to either select the most salient node in a graph with high structural importance by ranking the probability scores or measure the relatedness of two graphs by computing the similarity between two random walk distributions.

Applications Random walk algorithms have been applied in various NLP applications including measures of semantic similarity of texts (Ramage *et al.*, 2009) and semantic distance on semantic networks (Hughes and Ramage, 2007), word-sense disambiguation (Mihalcea, 2005; Tarau *et al.*, 2005), name disambiguation (Minkov *et al.*, 2006), query expansion (Collins-Thompson and Callan, 2005), keyword extraction (Mihalcea and Tarau, 2004), and cross-language information retrieval (Monz and Dorr, 2005). For example, given a semantic network and a word pair, Hughes and Ramage (2007) computed a word-specific stationary distribution using a random walk algorithm, and measured the distance between two words as the similarity between the random walk distributions on this graph, biased on each input word in a given word pair. To solve a name disambiguation task on email data, Minkov *et al.* (2006) built a graph of email-specific items (e.g., sender, receiver and subject) from a corpus of emails, and proposed a "lazy" topic-sensitive random walk algorithm which introduces a probability that the random walk would stop at a given node. Given an email graph and an ambiguous name appearing in an input email, a random walk is performed biased toward the text of the given email,

and the name is resolved to the correct reference by choosing the person node that has the highest score in the stationary distribution after convergence. To solve the keyword extraction task, Mihalcea and Tarau (2004) proposed to perform a random walk on a co-occurrence graph of words, and rank the importance of the words in the text based on their probability scores in the stationary distribution.

Graph Clustering Algorithms

Approach Common graph clustering algorithms include spectral clustering, random walk clustering and min-cut clustering. Spectral clustering algorithms make use of the spectrum (eigenvalues) of the Laplacian matrix of the graph to perform dimensionality reduction before conducting clustering using existing algorithms like K-means. Random walk clustering algorithms operate by conducting a t-step random walk on the graph, as a result, each node is represented as a probability vector indicating the t-step generation probabilities to all of the other nodes in the graph. Any clustering algorithm can be applied on the generation-link vectors. Note that for graph clustering purposes, a small value of t is more preferred because we are more interested in capturing the local structural information instead of the global structural information (encoded by the stationary distribution after a random walk converges). The min-cut algorithms can also be used to partition the graph into clusters.

Applications Graph clustering algorithms have been successfully applied to solve the text clustering task. For instance, Erkan (2006) proposed to use the n-dim probabilistic distribution derived from a t-step random walk on a directed generation graph (containing n document nodes) as the vector representation of each document in a corpus. Then these document representations can be consumed by a graph clustering algorithm to generate document clusters. Note that the generation graph is constructed by computing the generation probability of each ordered document pair in the corpus following the language-model approach proposed by Ponte and Croft (1998).

Graph Matching Algorithms

Approach Graph matching algorithms aim to compute the similarity between two graphs. Among them, Graph Edit Distance is the most commonly

used method to measure the dissimilarity of two graphs. It computes the distance as the number of changes (i.e., add, delete, substitute) needed to transform one graph into the other. Then the dissimilarity score can be converted into the similarity score.

Applications Graph matching algorithms have applications in the textual entailment task that aims at deciding whether a given sentence can be inferred from text. For example, Haghighi *et al.* (2005) assumed that a hypothesis is entailed from the text when the cost of matching the hypothesis graph to the text graph is low, and thus applied a graph matching algorithm to solve the problem.

Label Propagation Algorithms

Approach Label propagation algorithms (LPAs) is a class of semi-supervised graph-based algorithms that propagate labels from labeled data points to previously unlabeled data points. Basically, LPAs operate by propagating and aggregating labels iteratively across the graph. At each iteration, each node changes its label based on the labels that its neighboring nodes possess. As a result, the label information diffuses through the graph.

Applications LPA have been widely used in the network science literature for discovering community structures in complex networks. In the literature of NLP, LPA have been successfully applied in word-sense disambiguation (Niu *et al.*, 2005) and sentiment analysis (Goldberg and Zhu, 2006). These applications usually focus on the semi-supervised learning setting where labeled data is scarce, and leverage the LPA algorithm for propagating labels from limited labeled examples to a large amount of similar unlabeled examples with the assumption that similar examples should have similar labels.

Limitations and Connections to GNNs

Although traditional graph-based algorithms have been successfully applied in many NLP tasks, they have several limitations. First of all, they have limited expressive power. They mostly focus on capturing the structural information of graphs but do not consider the node and edge features which are also

very important for many NLP applications. Secondly, there is not a unified learning framework for traditional graph-based algorithms. Different graph-based algorithms have very different properties and settings, and are only suitable to some specific use cases.

The above limitations of traditional graph-based algorithms call for a unified graph-based learning framework with strong expressive power on modeling both the graph structures and node/edge properties. Recently, GNNs have gained increasing attention as a special class of neural networks which can model arbitrary graph-structured data. Most GNN variants can be regarded as a message passing based learning framework. Unlike traditional message passing based algorithms like LPA which operates by propagating labels across a graph, GNNs typically operate by transforming, propagating and aggregating nodes/edge features through several neural layers so as to learn better graph representations. As a general graph-based learning framework, GNNs can be applied to various graph-related tasks such as node classification, link prediction and graph classification.

0.3 Graph Neural Networks

In the previous chapter, we have illustrated various conventional graph-based methods for different NLP applications. In this chapter, we will elaborate basic foundations and methodologies for graph neural networks (GNNs) which are a class of modern neural networks which directly operate on graph-structured data (Wu *et al.*, 2022). To facilitate the description of the technologies, we list all the notations used throughout this survey in Table 1, which includes variables and operations in the domain of both the graph neural networks and NLP.

Table 1: Notation

Graph Basics	
A graph	$\mathcal{G}$
Edge set	$\mathcal{E}$
Vertex (node) set	$\mathcal{V}$

The number of vertexes (nodes)	n
The number of edges	m
A single vertex(node) $v_i \in \mathcal{V}$	v_i
A single edge $e_{i,j}$(connecting vertex v_i and vertex $v_j \in \mathcal{E}$	$e_{i,j}$
The neighbours of a vertex (node) v_i	$N(v_i)$
Adjacent matrix of a graph	A
Laplacian matrix	L
Diagonal degree matrix	D
The initial attributes of vertex $v_i \in \mathcal{V}$	$\mathbf{x}_i$
The initial attributes of edge $e_{i,j} \in \mathcal{E}$	$\mathbf{r}_{i,j}$
The embedding of vertex $v_i \in \mathcal{V}$	$\mathbf{h}_i$
The embedding of edge $e_{i,j} \in \mathcal{E}$	$\mathbf{e}_{i,j}$
NLP Basics	
Vocabulary	V
Source language	s
Target language	t
Corpus of words/aligned sentences used for training	C
The i^{th} word in corpus C	w_i
The embedding of word w_i	$\mathbf{w}_i$
The embedding vector's dimensionality	d
The number of words	n
The i^{th} document in source(target) language	$doc_i^s(doc_i^t)$
Representation of document $doc_i^s(doc_i^t)$	$\mathbf{d}_i^s(\mathbf{d}_i^t)$
The i^{th} paragraph in source(target) language	$para_i^s(para_i^t)$
Representation of paragraph $para_i^s(para_i^t)$	$\mathbf{p}_i^s(\mathbf{p}_i^t)$
The i^{th} sentence in source(target) language	$sent_i^s(sent_i^t)$
Representation of sentence $sent_i^s(sent_i^t)$	$\mathbf{s}_i^s(\mathbf{s}_i^t)$

0.3.1 Foundations

Graph neural networks are essentially graph representation learning models and can be applied to node-focused tasks and graph-focused tasks. GNNs learn embeddings for each node in the graph and aggregate the node embeddings to produce the graph embeddings. Generally, the learning process of node embeddings utilizes graph structure and input node embeddings, which can be summarized as:

$$\mathbf{h}_i^{(l)} = f_{\mathbf{filter}}(A, \mathbf{H}^{(l-1)}) \tag{1}$$

where $A \in \mathbb{R}^{n \times n}$ is the adjacency matrix of the graph, $\mathbf{H}^{(l-1)} = \{\mathbf{h}_1^{(l-1)}, \mathbf{h}_2^{(l-1)}, ..., \mathbf{h}_n^{(l-1)}\} \in \mathbb{R}^{n \times d}$ denotes the input node embeddings at the $l-1$-th GNN layer, and $\mathbf{H}^{(l)}$ is the updated node embeddings. d is the dimension of $\mathbf{h}_i^{(l-1)}$. We refer to the process depicted in Eq.(1) as *graph filtering* and $f_{\mathbf{filter}}(\cdot, \cdot)$ is named as a graph filter. The specific models then differ only in how $f_{\mathbf{filter}}(\cdot, \cdot)$ is chosen and parameterized. Graph filtering does not change the structure of graph, but refines the node embeddings. Graph filtering layers are stacked to L layers to generate final node embeddings.

Since graph filtering does not change the graph structure, pooling operations are introduced to aggregate node embeddings to generate graph-level embeddings inspired by CNNs. In GNN models, the *graph pooling* takes a graph and its node embeddings as inputs and then generates a smaller graph with fewer nodes and its corresponding new node embeddings. The graph pooling operation can be summarized as follows:

$$A', \mathbf{H}' = f_{\mathbf{pool}}(A, \mathbf{H}) \tag{2}$$

where $f_{\mathbf{pool}}(\cdot, \cdot)$ $A \in \mathbb{R}^{n \times n}$ and $A' \in \mathbb{R}^{n' \times n'}$ are the adjacency matrices before and after graph pooling. $\mathbf{H} \in \mathbb{R}^{n \times d}$ and $\mathbf{H}' \in \mathbb{R}^{n' \times d'}$ are the node embeddings before and after graph pooling. n' is set to be 1 in most cases to get the embedding for the entire graph.

0.3.2 Methodologies

Graph Filtering

There exists a variety of implementations of graph filter f in Eq.(1), which could be roughly categorized into spectral-based graph filters, spatial-based graph filters, attention-based graph filters and recurrent-based graph filters.

Conceptually, the spectral-based graph filters are based on spectral graph theory while the spatial-based methods compute a node embedding using its spatially close neighbor nodes on the graph. Some spectral-based graph filters can be converted to spatial-based graph filters. The attention-based graph filters are inspired by the self-attention mechanism (Vaswani *et al.*, 2017) to assign different attention weights to different neighbor nodes. Recurrent-based graph filters introduce gating mechanism, and the model parameters are shared across different GNN layers. Next, we will explain these four types of graph filters in detail by introducing some of their representative GNN models.

Spectral-based Graph Filters Inspired by graph signal processing, Defferrard *et al.* proposed a spectral graph theoretical formulation of CNNs, which generalizes CNNs to graphs and provides the same linear computational complexity and constant learning complexity as classical CNNs. A more typical example of spectral-based graph filters is Graph Convolutional Networks (GCN) (Kipf and Welling, 2016). Spectral convolution on graphs is defined as the multiplication of a signal $\mathbf{x}_i \in \mathbb{R}^n$ (a scalar for node v_i) with the filter $f_{\mathbf{filter}} = \text{diag}(\theta)$ parameterized by $\theta \in \mathbb{R}^n$ in the Fourier domain:

$$f_{\mathbf{filter}} * \mathbf{x}_i = \mathbf{U} f(\mathbf{\Lambda}) \mathbf{U}^T \mathbf{x}_i \tag{3}$$

where $\mathbf{U}$ is the matrix of eigenvectors of the normalized graph Laplacian $L = I_n - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$. I_n is the identity matrix, D is the degree matrix and $\mathbf{\Lambda}$ is the eigenvalues of L.

However, the computation of the full eigen-decomposition is prohibitively expensive. To solve this problem, Defferrard *et al.* (2016) uses a truncated expansion in terms of Chebyshev polynomials $\mathbf{T}_p(x)$ up to P^{th}-order to approximate $\mathbf{g}_\theta(\mathbf{\Lambda})$. Eq. (3) can be represented as follows:

$$f'_{\mathbf{filter}} * \mathbf{x}_i \approx \sum_{p=0}^{P} \theta'_p \mathbf{T}_p(\tilde{\mathbf{L}}) \mathbf{x}_i \tag{4}$$

where $\tilde{L} = \frac{2}{\lambda_{max}} L - I_n$. λ_{max} is the largest eigenvalue of L. $\theta'_k \in \mathbb{R}^P$ is a vector of Chebyshev coefficients. The Chebyshev polynomials can be defined recursively: $\mathbf{T}_k(\mathbf{x}_i) = 2\mathbf{x}_i \mathbf{T}_{k-1}(\mathbf{x}_i) - \mathbf{T}_{k-2}(\mathbf{x}_i)$, with $\mathbf{T}_0(\mathbf{x}_i) = 1$ and $\mathbf{T}_1(\mathbf{x}_i) = \mathbf{x}_i$. Eq.(4) is a Kth-order polynomial in the Laplacian, which shows that every central node depends only on nodes in the P-hop range.

Therefore, a neural network model based on graph convolution can stack multiple convolutional layers using Eq. (4). By limiting the layer-wise convolution operation to $P = 1$ and stacking multiple convolutional layers, Kipf and Welling (2016) proposed a multi-layer Graph Convolutional Network (GCN). It further approximates $\lambda_{max} \approx 2$ and Eq. (4) is simplified to:

$$f'_{\mathbf{filter}} * \mathbf{h}_i^{(l)} \approx \theta'_0 \mathbf{h}_i^{(l)} + \theta'_1 (L - I_n)\mathbf{h}_i^{(l)} = \theta'_0 \mathbf{h}_i^{(l)} - \theta'_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \mathbf{h}_i^{(l)} \quad (5)$$

with two free parameters θ'_0 and θ'_1. To alleviate the problem of overfitting and minimize the number of operations (such as matrix multiplications), it is beneficial to constrain the number of parameters by setting a single parameter $\theta = \theta'_0 = -\theta'_1$:

$$f_{\mathbf{filter}} * \mathbf{h}_i^{(l)} \approx \theta (I_n + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}) \mathbf{h}_i^{(l)} \quad (6)$$

Repeat application of this operator may cause numerical instability and explosion/vanishing gradients, Kipf and Welling (2016) proposed to use a *renormalization trick*: $I_n + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \rightarrow \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$, with $\tilde{A} = A + I_n$ and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. Finally, the definition can be generalized with a signal $\mathbf{H} \in \mathbb{R}^{n \times d}$ with d input channels (i.e. a d-dimensional feature vector for each node) and F filters or feature maps as follows:

$$\mathbf{H}^{(l)} = \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} \mathbf{H}^{(l-1)} \mathbf{W}^{(l-1)}) \quad (7)$$

Here, $\mathbf{W}^{(l-1)}$ is a layer-specific trainable weight matrix and $\sigma(\cdot)$ denotes an activation function. $\mathbf{H}^{(l)} \in \mathbb{R}^{n \times d}$ is the activated node embeddings at $(l-1)$-th layer.

Spatial-based Graph Filters Analogous to the convolutional operation of a conventional CNN, spatial-based graph filters operate the graph convolutions based on a node's spatial relations. The spatial-based graph filters derive the updated representation for the target node via convolving its representation with its neighbors' representations. On the other hand, spatial-based graph filters hold the idea of information propagation, namely, message passing. The spatial-based graph convolutional operation essentially propagates node information as messages along the edges. Here we introduce two typical GNNs based on spatial-based graph filters are Message Passing Neural Network (MPNN) (Gilmer *et al.*, 2017) and GraphSage (Hamilton *et al.*, 2017a).

MPNN (Gilmer *et al.*, 2017) proposes a general framework of spatial-based graph filters $f_{\mathbf{filter}}$ which is a composite function consisting of f_U and f_M. It treats graph convolutions as a message passing process in which information can be passed from one node to another along the edges directly. MPNN runs K-step message passing iterations to let information propagate further to K-hop neighboring nodes. The message passing function, namely the spatial-based graph filter, on the target node v_i is defined as

$$\mathbf{h}_i^{(l)} = f_{\mathbf{filter}}(A, \mathbf{H}^{(l-1)}) = f_U(\mathbf{h}_i^{(l-1)}, \sum_{v_j \in N(v_i)} f_M(\mathbf{h}_i^{(l-1)}, \mathbf{h}_j^{(l-1)}, \mathbf{e}_{i,j})), \tag{8}$$

where $\mathbf{h}_i^{(0)} = \mathbf{x}_i$, $f_U(\cdot)$ and $f_M(\cdot)$ are the update and message aggregate functions with learnable parameters, respectively. After deriving the hidden representations of each node, $\mathbf{h}_i^{(L)}$ (L is the number of graph convolution layers) can be passed to an output layer to perform node-level prediction tasks or to a readout function to perform graph-level prediction tasks. MPNN is very general to include many existing GNNs by applying different functions of $f_U(\cdot)$ and $f_M(\cdot)$.

Considering that the number of neighbors of a node can vary from one to a thousand or even more, it is inefficient to take the full size of a node's neighborhood in a giant graph with thousands of millions of nodes. GraphSage (Hamilton *et al.*, 2017a) adopts sampling to obtain a fixed number of neighbors for each node as

$$f_{\mathbf{filter}}(A, \mathbf{H}^{(l-1)}) = \sigma(\mathbf{W}^{(l)} \cdot f_M(\mathbf{h}_i^{(l-1)}, \{\mathbf{h}_j^{(l-1)}, \forall v_j \in N(v_i)\})), \tag{9}$$

where $N(v_i)$ is a random sample of the neighboring nodes of node v_i. The aggregation function can be any functions that are invariant to the permutations of node orderings such as mean, sum or max operations.

Attention-based Graph Filters The original versions of GNNs take edge connections of the input graph as fixed, and do not dynamically adjust the connectivity information during the graph learning process. Motivated by the above observation, and inspired by the successful applications of multi-head attention mechanism in the Transformer model (Vaswani *et al.*, 2017; Velickovic *et al.*, 2018) proposed the Graph Attention Network (GAT) by introducing the multi-head attention mechanism to the GNN architecture which is able

to dynamically learn the weights (i.e., attention scores) on the edges when performing message passing. More specifically, when aggregating embeddings from neighboring nodes for each target node in the graph, the semantic similarity between the target node and each neighboring node will be considered by the multi-head attention mechanism, and important neighboring nodes will be assigned higher attention scores when performing the neighborhood aggregation. For the l-th layer, GAT thus uses the following formulation of the attention mechanism,

$$\alpha_{ij} = \frac{\exp(\text{LeakyReLU}(\mathbf{u}^{(l)^T}[\mathbf{W}^{(l)}\mathbf{h}_i^{(l-1)}||\mathbf{W}^{(l)}\mathbf{h}_j^{(l-1)}]))}{\sum_{v_k \in N(v_i)} \exp(\text{LeakyReLU}(\mathbf{u}^{(l)^T}[\mathbf{W}^{(l)}\mathbf{h}_i^{(l-1)}||\mathbf{W}^{(l)}\mathbf{h}_k^{(l-1)}]))} \tag{10}$$

where $\mathbf{u}^{(l)}$ and $\mathbf{W}^{(l)}$ are the weight vector and weight matrix at l-th layer, respectively, and $||$ is the vector concatenation operation. Note that $N(v_i)$ is the 1-hop neighborhood of v_i including itself. After obtaining the attention scores α_{ij} for each pair of nodes v_i and v_j, the updated node embeddings can be computed as a linear combination of the input node features followed by some nonlinearity σ, formulated as,

$$\mathbf{h}_i^{(l)} = f_{\textbf{filter}}(A, \mathbf{H}^{(l-1)}) = \sigma(\sum_{v_j \in N(v_i)} \alpha_{ij}\mathbf{W}^{(l)}\mathbf{h}_j^{(l-1)}) \tag{11}$$

In order to stabilize the learning process of the above self-attention, inspired by Vaswani *et al.* (2017), multiple independent self-attention mechanisms are employed and their outputs are concatenated to produce the following node embedding:

$$f_{\textbf{filter}}(A, \mathbf{H}^{(l-1)}) = ||_{k=1}^{K}\sigma(\sum_{v_j \in N(v_i)} \alpha_{ij}^k\mathbf{W}_k^{(l)}\mathbf{h}_j^{(l-1)}), \tag{12}$$

while the final GAT layer (i.e., the L-th layer for a GNNs with L layers) employs averaging instead of concatenation to combine multi-head attention outputs.

$$f_{\textbf{filter}}(A, \mathbf{H}^{(L-1)}) = \sigma(\frac{1}{K}\sum_{k=1}^{K}\sum_{v_j \in N(v_i)} \alpha_{ij}^k\mathbf{W}_k^{(L)}\mathbf{h}_j^{(L-1)}) \tag{13}$$

Recurrent-based Graph Filters A typical example of recurrent-based graph filters is the Gated Graph Neural Networks (GGNN)-filter. The biggest

modification from typical GNNs to GGNNs is the use of Gated Recurrent Units (GRU) (Cho *et al.*, 2014). Analogous to RNN, GGNN unfolds the recurrence in a fixed T time steps and uses back propagation through time to calculate the gradients. The GGNN-filter also takes the edge type and edge direction into consideration. To this end, $e_{i,j}$ denotes the directed edge from node v_i to node v_j and the edge type of $e_{i,j}$ is $t_{i,j}$. The propagation process of recurrent-based filter $f_{\mathbf{filter}}$ in GGNN can be summarized as follows:

$$\mathbf{h}_i^{(0)} = [\mathbf{x}_i^T, \mathbf{0}]^T \tag{14}$$

$$\mathbf{a}_i^{(l)} = A_{i:}^T [\mathbf{h}_1^{(l-1)} ... \mathbf{h}_n^{(l-1)}]^T \tag{15}$$

$$\mathbf{h}_i^{(l)} = \text{GRU}(\mathbf{a}_i^{(l)}, \mathbf{h}_i^{(l-1)}) \tag{16}$$

where $A \in \mathbb{R}^{dn \times 2dn}$ is a matrix determining how nodes in the graph communicating with each other. n is the number of nodes in the graph. $A_{i:} \in \mathbb{R}^{d \times 2d}$ are the two columns of blocks in A corresponding to node v_i. In Eq. (14), the initial node feature $\mathbf{x}_i$ are padded with extra zeros to make the input size equal to the hidden size. Eq. (15) computes $\mathbf{a}_i^{(l)} \in \mathbb{R}^{2d}$ by aggregating information from different nodes via incoming and outgoing edges with parameters dependent on the edge type and direction. The following step uses a GRU unit to update the hidden state of node v by incorporating $\mathbf{a}_i^{(l)}$ and the previous timestep hidden state $\mathbf{h}_i^{(l-1)}$.

Graph Pooling

Graph pooling layers are proposed to generate graph-level representations for graph-focused downstream tasks, such as graph classification and prediction based on the node embedding learned from the graph filtering. This is because the learned node embeddings are sufficient for node-focused tasks, however, for graph-focused tasks, a representation of the entire graph is required. To this end, we need to summarize the node embeddings information and the graph structure information. The graph pooling layers can be classified into two categories: flat graph pooling and hierarchical graph pooling. The flat graph pooling generates the graph-level representation directly from the node embeddings in a single step. In contrast, the hierarchical graph pooling contains several graph pooling layers and each of the pooling layer follows a stack of graph filters. In this section, we briefly introduce several representative flat pooling layers and hierarchical pooling layers.

Flat Graph Pooling Ideally, an aggregator function would be invariant to permutations of its input while maintaining a large expressive capacity. The graph pooling operation $f_{\mathbf{pool}}$ is commonly implemented as Max-pooling and Average-pooling. Another popular choices are the variants of the Max-pooling and Average pooling operations by following a fully-connected layer (FC) transformation. The resulting max pooling and FCmax can be expressed as:

$$\mathbf{r}_i = \max(\mathbf{H}_{:,i}) \;\; or \;\; \mathbf{r}_i = \max(\mathbf{W}\mathbf{H}_{:,i}) \tag{17}$$

where i denotes the i-th channel of the node embedding and $\mathbf{H}_{:,i} \in \mathbf{R}^{n\times 1}$ is a vector. $\mathbf{W}$ is a matrix that denotes to the trainable parameters of the FCmax pooling layer. $\mathbf{r}_i$ is a scalar and the final graph embedding $\mathbf{R} = [r_1, r_2, ..., r_n]^T$. Finally, a powerful but less common pooling operation is the BiLSTM aggregation function which is not permutation invariant on the set of node embeddings. However, it has been often demonstrated to have better expressive power than other flat pooling operations (Hamilton *et al.*, 2017a; Zhang *et al.*, 2019c).

Hierarchical Graph Pooling Hierarchical graph pooling coarsens the graph step by step to learn the graph-level embeddings. Hierarchical pooling layers can be divided into two categories according to the ways to coarsen the graph. One type of hierarchical pooling layer coarsens the graph by sub-sampling the most important nodes as the nodes of the coarsened graph (Gao *et al.*, 2019). Another type of hierarchical pooling layer combines nodes in the input graph to form supernodes, which serve as the nodes of the coarsened graph (Ying *et al.*, 2018; Ma *et al.*, 2019). After sub-sampling nodes or generating supernodes, the hierarchical graph pooling $f_{\mathbf{pool}}$ can be summarized as: (1) generating graph structure for the coarsened graph; (2) generating node features for the coarsened graph. The graph structure for the coarsened graph is generated from the input graph:

$$A' = COARSEN(A) \tag{18}$$

where $A \in \mathbb{R}^{n\times n}$ is the adjacent matrix of the input graph, and $A' \in \mathbb{R}^{n'\times n'}$ is the adjacent matrix of the coarsened graph. $f(.)$ is the graph sub-sampling or supernodes generating function.

0.4 Graph Construction Methods for NLP

In the previous section, we have discussed the basic foundations and methods of GNNs once given a graph input. Unfortunately, for most of the NLP tasks, the typical inputs are sequence of text rather than graphs. Therefore, how to construct a graph input from sequences of text becomes a demanding step in order to leverage the power of GNNs. In this chapter, we will focus on introducing two major graph construction approaches, namely static graph construction and dynamic graph construction for constructing graph structured inputs in various NLP tasks.

0.4.1 Static Graph Construction

The static graph construction approach aims to construct the graph structures during preprocessing typically by leveraging existing relation parsing tools (e.g., dependency parsing) or manually defined rules. Conceptually, a static graph incorporates different domain/external knowledge hidden in the original text sequences, which augments the raw text with rich structured information.

In this subsection, we summarize various static graph construction methods in the GNN for NLP literature and group them into totally eleven categories, as shown in Table 2. We assume that the input is a document $doc = \{para_1, para_2, ..., para_n\}$, which consists of n paragraph denoted as $para$. Similarly, a paragraph consisting of m sentences is denoted as $para_i = \{sent_1, sent_2, ..., sent_m\}$. Each sentence then consists of l words denoted as $sent_i = \{w_1, w_2, ..., w_l\}$.

Static Graph Construction Approaches

Dependency Graph Construction The dependency graph is widely used to capture the dependency relations between different objects in the given sentences. Formally, given a paragraph, one can obtain the dependency parsing tree (e.g., syntactic dependency tree or semantic dependency parsing tree) by using various NLP parsing tools (e.g., Stanford CoreNLP (Lee *et al.*, 2011)). Then one may extract the dependency relations from the dependency parsing tree and convert them into a dependency graph (Xu *et al.*, 2018b; Song *et al.*, 2018c). Moreover, since the given paragraph has sequential information while the graph nodes are unordered, one may introduce the sequential links to

Table 2: Two major graph construction approaches: static and dynamic graph constructions

Approaches	Techniques		References
Static Graph	Dependency Graph		Zhang *et al.* (2019a), Guo *et al.* (2019b), Zhang and Qian (2020), and Fei *et al.* (2020) Bastings *et al.* (2017), Nguyen and Grishman (2018), Ji *et al.* (2019), and Liu *et al.* (2018b) Xu *et al.* (2018c), Zhang *et al.* (2018c), Song *et al.* (2018e), and Li *et al.* (2017) Do and Rehbein (2020), Yan *et al.* (2019), Marcheggiani *et al.* (2018), and Zhou *et al.* (2020b) Vashishth *et al.* (2018), Xia *et al.* (2020), Jin *et al.* (2020a), and Huang and Carley (2019) Sahu *et al.* (2019), Cui *et al.* (2020c), Xu *et al.* (2020c), and Zhang *et al.* (2020a) Liu *et al.* (2019b), Li *et al.* (2020a), Wang *et al.* (2020b), and Tang *et al.* (2020a) Qian *et al.* (2019), Pouran Ben Veyseh *et al.* (2020), and Wang *et al.* (2020c)
	Constituency Graph		Li *et al.* (2020a), Marcheggiani and Titov (2020), and Xu *et al.* (2018c)
	AMR Graph		Liao *et al.* (2018), Wang *et al.* (2020f), Wang *et al.* (2020g), and Ribeiro *et al.* (2019b) Jin and Gildea (2020), Jin *et al.* (2020a), and Cai and Lam (2020b) Bai *et al.* (2020), Beck *et al.* (2018a), and Yao *et al.* (2020) Zhang *et al.* (2020d), Zhao *et al.* (2020b), and Zhu *et al.* (2019b) Song *et al.* (2020), Song *et al.* (2018d), Song *et al.* (2019), and Damonte and Cohen (2019)
	Information Extraction Graph		Wu *et al.* (2020c) and Vashishth *et al.* (2018) Huang *et al.* (2020b) and Gupta *et al.* (2019)
	Discourse Graph		Song *et al.* (2018e), Li *et al.* (2020b), Yasunaga *et al.* (2017), and Xu *et al.* (2020a)
	Knowledge Graph		Ye *et al.* (2019), Yang *et al.* (2019), Gupta *et al.* (2019), and Xu *et al.* (2020b) Sun *et al.* (2020b), Xu *et al.* (2019a), and Wang *et al.* (2019f) Kapanipathi *et al.* (2020), Zhang *et al.* (2019d), Zhang *et al.* (2020g), and Sun *et al.* (2018a) Malaviya *et al.* (2020), Huang *et al.* (2020b), Schlichtkrull *et al.* (2018), and Sun *et al.* (2019b) Bansal *et al.* (2019), Saxena *et al.* (2020), and Koncel-Kedziorski *et al.* (2019) Teru *et al.* (2020), Lin *et al.* (2019a), Ghosal *et al.* (2020), and Feng *et al.* (2020a) Wu *et al.* (2019b), Wu *et al.* (2019a), Wu *et al.* (2020c), and Wu *et al.* (2020b) Wang *et al.* (2019a), Wang *et al.* (2019c), Wang *et al.* (2020h), and Wang *et al.* (2019g) Zhao *et al.* (2020c), Shang *et al.* (2019), Jin *et al.* (2019), and Nathani *et al.* (2019a) Sorokin and Gurevych (2018b), Cao *et al.* (2019b), Han *et al.* (2020), and Xie *et al.* (2020)
	Coreference Graph		Sahu *et al.* (2019), Qian *et al.* (2019), Xu *et al.* (2020c), and Xu *et al.* (2020a) De Cao *et al.* (2018) and Luan *et al.* (2019)
	Topic Graph		Linmei *et al.* (2019) and Li *et al.* (2020b)
	Similarity Graph Construction		Xia *et al.* (2019), Yao *et al.* (2019b), and Yasunaga *et al.* (2017) Linmei *et al.* (2019), Zhou *et al.* (2020a), and Wang *et al.* (2020a) Liu *et al.* (2019a), Hu *et al.* (2020b), Jia *et al.* (2020), and Li *et al.* (2020b)
	Co-occurrence Graph		Christopoulou *et al.* (2019), Zhang and Qian (2020), and Hu *et al.* (2020b) Zhang *et al.* (2020f), Yao *et al.* (2019b), and De Cao *et al.* (2018) Edouard *et al.* (2017), Zhu *et al.* (2018), and Liu *et al.* (2019a)
	App-driven Graph		Ding *et al.* (2019b), Yin *et al.* (2020), Luo and Zhao (2020), and Ding *et al.* (2019a) Sui *et al.* (2019), Tang *et al.* (2020c), Ran *et al.* (2019), and Hu *et al.* (2019e) Gui *et al.* (2019), Li and Goldwasser (2019), Xiao *et al.* (2019), and Xu *et al.* (2018a) Qu *et al.* (2020), Bogin *et al.* (2019b), Huo *et al.* (2019), and Shao *et al.* (2020) Fernandes *et al.* (2019), Liu *et al.* (2020), Huang *et al.* (2019), and Linmei *et al.* (2019) Bogin *et al.* (2019a), LeClair *et al.* (2020), Qiu *et al.* (2019), and Zheng *et al.* (2020) Ferreira and Freitas (2020), Zheng and Kordjamshidi (2020), and Fang *et al.* (2020a) Allamanis *et al.* (2018), Christopoulou *et al.* (2019), and Thayaparan *et al.* (2019)
Dynamic graph	Graph Similarity Metric Learning	Node Embedding Based	Chen *et al.* (2020h), Chen *et al.* (2020i), Chen *et al.* (2020f), and Chen *et al.* (2020a)
		Structure-aware	Liu *et al.* (2019c) and Liu *et al.* (2021b)
	Graph Sparsification Techniques		Chen *et al.* (2020h), Chen *et al.* (2020i), and Chen *et al.* (2020f)
	Combining Intrinsic and Implicit Graph Structures		Chen *et al.* (2020f) and Liu *et al.* (2021b)

reserve such vital information in the graph structure (Sahu *et al.*, 2019; Qian *et al.*, 2019; Xu *et al.*, 2018c; Li *et al.*, 2017). Next, we will discuss a representative dependency graph construction method given the inputs $para$ and its extracted parsing tree, including three key steps: 1) constructing dependency relation, 2) constructing sequential relation, and 3) final graph conversion. An example for the dependency graph is shown in Fig. 2.

Step 1: Dependency Relations. Given the sentences in a specific paragraph, one first obtains the dependency parsing tree for each sentence. We denote dependency relations in the dependency tree as $(w_i, rel_{i,j}, w_j)$, where w_i,

w_j are the word nodes linked by an edge type $rel_{i,j}$. Conceptually, an edge denotes a dependency relation "w_i depends on w_j with relation $rel_{i,j}$". We define the dependency relation set as $\mathcal{R}_{dep}$.

Step 2: Sequential Relations. The sequential relation encodes the adjacent relation of the elements in the original paragraph. Specifically, for dependency graph constructing, we define the sequential relation set $\mathcal{R}_{seq} \subseteq \mathcal{V} \times \mathcal{V}$, where $\mathcal{V}$ is the basic element (i.e., word) set. For each sequential relation $(w_i, w_{i+1}) \in \mathcal{R}_{seq}$, it means w_i is adjacent to w_{i+1} in the given paragraph.

Step 3: Dependency Graph. The dependency graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ consists of the word nodes and two relations discussed above. Given the paragraph $para$, dependency relation set $\mathcal{R}_{dep}$, and the sequential relation set $\mathcal{R}_{seq}$, firstly, for each relation $(w_i, rel_{i,j}, w_j) \in \mathcal{R}_{dep}$, one adds the nodes v_i (for the word w_i) and v_j (for the word w_j) and a directed edge from node v_i to node v_j with edge type $rel_{i,j}$. Secondly, for each relation $(w_i, w_j) \in \mathcal{R}_{seq}$, one adds two nodes v_i (for the word w_i) and v_j (for the word w_j) and an undirected edge between nodes v_i and v_j with specific sequential type.

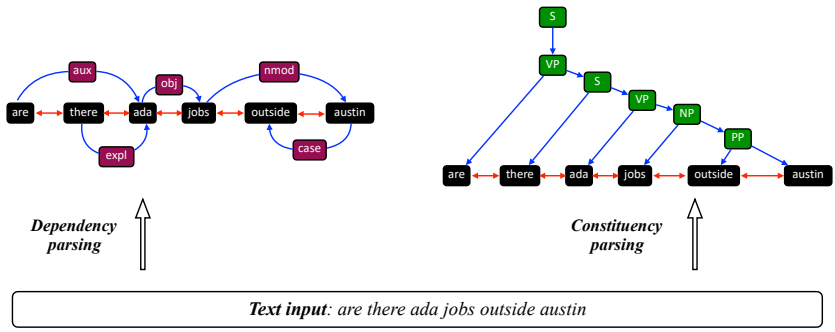


Figure 2: An example is shown for the dependency graph (left) and the constituency graph (right), respectively. The text input is from **JOBS640** (Luke, 2005) dataset.

Constituency Graph Construction The constituency graph is another widely used static graph that is able to capture phrase-based syntactic relations in one or more sentences. Unlike dependency parsing, which only focuses on one-to-one correspondences between single words (i.e., word level), constituency parsing models the assembly of one or several corresponded words (i.e., phrase level). Thus it provides a new insight about the grammatical struc-

ture of a sentence. In this following subsection, we will discuss the typical approach for constructing a constituency graph (Li *et al.*, 2020a; Marcheggiani and Titov, 2020; Xu *et al.*, 2018c). We first explain the basic concepts of the constituency relations and then illustrate the constituency graph construction procedure. An example for the Constituency graph is shown in Fig. 2.

Step 1: Constituency Relations. In linguistics, constituency relation means the relation following the phrase structure grammars instead of the dependency relation and dependency grammars. Generally, the constituency relation derives from the subject(noun phrase NP)-predicate(verb phrase VP) relation. In this part, we only discuss the constituency relation deriving from the constituency parsing tree. Unlike the dependency parsing tree, in which all nodes have the same type, the constituency parsing tree distinguishes between the terminal and non-terminal nodes. Non-terminal categories of the constituency grammar label the parsing tree's interior nodes (e.g., S for sentence, and NP for noun phrase). In contrast, the leaf nodes are labeled by terminal categories (words in sentences). The nodes set can be denoted as: 1) non-terminal nodes set $\mathcal{V}_{nt}$ (e.g. S and NP) and 2) terminal nodes set $\mathcal{V}_{words}$. The constituency relation set are associated with the tree's edges, which can be denoted as $\mathcal{R}_{cons} \subseteq \mathcal{V}_{nt} \times (\mathcal{V}_{nt} + \mathcal{V}_{words})$.

Step 2: Constituency Graph. A constituency graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ consists of both the non-terminal nodes $\mathcal{V}_{nt}$ and the terminal nodes $\mathcal{V}_{words}$, and the constituency edges as well as the sequential edges. Similar to the dependency graph, given a paragraph $para$ and the constituency relation set $\mathcal{R}_{cons}$, for each constituency relation $(w_i, rel_{i,j}, w_j) \in \mathcal{R}_{cons}$, one adds the nodes v_i (for the word w_i) and v_j (for the word w_j) and a directed edge from node v_i to node v_j. And then for each word nodes pair (v_i, v_j) for the words which are adjacent in the original text, one adds an undirected edge between them with the specific sequential type. These sequential edges are used to reserve the sequential information (Li *et al.*, 2020a; Xu *et al.*, 2018c).

AMR Graph Construction The AMR graphs are rooted, labeled, directed, acyclic graphs, which are widely used to represent the high-level semantic relations between abstract concepts of the unstructured and concrete natural text. Different from the syntactic idiosyncrasies, the AMR is the high-level semantic abstraction. More concretely, the different sentences that are semantically similar may share the same AMR parsing results, e.g., "Paul described

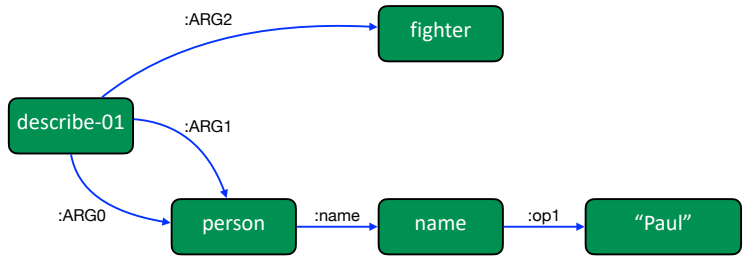


Figure 3: An example of AMR graph, the original sentence is "Paul's description of himself: a fighter".

himself as a fighter" and "Paul's description of himself: a fighter", as shown in Fig. 3. Despise the fact that the AMR is biased toward English, it is a powerful auxiliary representation for linguistic analysis (Song *et al.*, 2018d; Damonte and Cohen, 2019; Wang *et al.*, 2020f). Similar to the previously introduced dependency and constituency trees, an AMR graph is derived from an AMR parsing tree. Next, we focus on introducing the general procedure of constructing the AMR graph based on the AMR parsing tree. We will discuss the basic concept of AMR relation and then show how to convert the relations into an AMR graph.

Step 1: AMR Relations. Conceptually, there are two types of nodes in the AMR parsing tree: 1) the name (e.g. "Paul") is the specific value of the node instance and 2) the concepts are either English words (e.g. "boy"), PropBank framesets (Kingsbury and Palmer, 2002) (e.g. "want-01"), or special keywords. The name nodes are the unique identities, while the concept nodes are shared by different instances. The edges that connect nodes are called relations (e.g. :ARG0 and :name). One may extract these AMR relations from the node pairs with edges, which is denoted as $(n_i, r_{i,j}, n_j) \in \mathcal{R}_{amr}$.

Step 2: AMR Graph. The AMR graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, which is rooted, labeled, directed, acyclic graph (DAG), consists of the AMR nodes and AMR relations discussed above. Similar to the dependency and constituency graphs, given the sentence $sent$ and the AMR relation set $\mathcal{R}_{amr}$, for each relation $(n_i, r_{i,j}, n_j) \in \mathcal{R}_{amr}$, one adds the nodes v_i (for the AMR node n_i) and v_j (for the AMR node n_j) and add a directed edge from node v_i to node v_j with edge type $r_{i,j}$

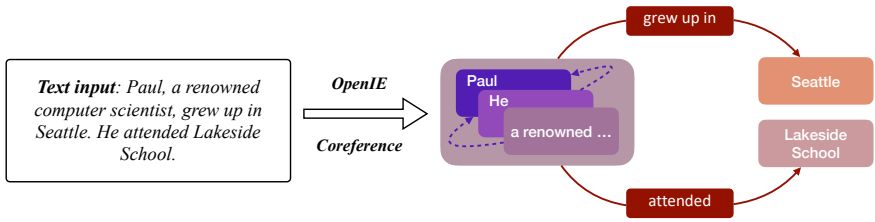


Figure 4: An example for IE graph construction which contains both the Co-reference process and the Open Information Extraction process.

Information Extraction Graph Construction The information extraction graph (IE Graph) aims to extract the structural information to represent the high-level information among natural sentences, e.g., text-based documents. These extracted relations that capture relations across distant sentences have been demonstrated helpful in many NLP tasks (Wu *et al.*, 2020c; Vashishth *et al.*, 2018; Gupta *et al.*, 2019). In what follows, we will discuss the technical details on how to construct an IE graph for a given paragraph $para$ (Huang *et al.*, 2020b; Vashishth *et al.*, 2018). We divide this process into two three basic steps: 1) coreference resolution, 2) constructing IE relations, and 3) graph construction.

Step 1: Coreference Resolution. Coreference resolution is the basic procedure for information extraction task which aims to find expressions that refer to the same entities in the text sequence (Huang *et al.*, 2020b). As shown in Figure 4, the name "Paul", the noun-term "He" and "a renowned computer scientist" may refer to the same object (person). Many NLP tools such as OpenIE (Angeli *et al.*, 2015) provide coreference resolution function to achieve this goal. We denotes the coreference cluster C as a set of phrases referring to the same object. Given a paragraph, one can obtain the coreference sets $\mathcal{C} = \{C_1, C_2, ..., C_n\}$ extracting from unstructured data.

Step 2: IE Relations. To construct an IE graph, the first step is to extract the triples from the paragraphs, which could be completed by leveraging some well-known information extraction systems (i.e. OpenIE (Angeli *et al.*, 2015)). We call each triple (subject, predicate, object) as a relation, which is denoted $(n_i, r_{i,j}, n_j) \in \mathcal{R}_{ie}$. It is worth noting if two triples differ only by one argument, and the other arguments overlap, one only keep the longer triple.

Step 3: IE Graph Construction. The IE graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ consists of the IE nodes

and IE relations discussed above. Given the paragraph $para$ and the IE relation set $\mathcal{R}_{ie}$, for each relation $(n_i, r_{i,j}, n_j) \in \mathcal{R}_{ie}$, one adds the nodes v_i (for the subject n_i) and v_j (for the object n_j) and add a directed edge from node v_i to node v_j with the corresponding predicate types (Huang *et al.*, 2020b). And then, for each coreference cluster $C_i \in \mathcal{C}$, one may collapse all coreferential phrases in C_i into one node. This could help greatly reduce the number of nodes and eliminate the ambiguity by keeping only one node.

Discourse Graph Construction Many NLP tasks suffer from long dependency challenge when the candidate document is too long. The discourse graph, which describes how two sentences are logically connected to one another, are proved effective to tackle such challenge (Christensen *et al.*, 2013). In the following subsection, we will briefly discuss the discourse relations between given sentences and then introduce the general procedure to construct the discourse graphs (Song *et al.*, 2018e; Li *et al.*, 2020b; Yasunaga *et al.*, 2017; Xu *et al.*, 2020a).

Step 1: Discourse Relation. The discourse relations derive from the discourse analysis, which aims to identify sentence-wise ordering constraints over a set of sentences. Given two sentences $sent_i$ and $sent_j$, one can define the discourse relation as $(sent_i, sent_j)$, which represents the discourse relation "sentence $sent_j$ can be placed after sentence $sent_i$." The discourse analysis has been explored for years, and many theories have been developed for modeling discourse relations such as the Rhetorical Structure Theory (RST) (Mann and Thompson, 1987) and G-Flow (Christensen *et al.*, 2013). In many NLP tasks, given a document doc, one firstly segments doc into sentences set $\mathcal{V} = \{sent_1, sent_2, ..., sent_m\}$. Then one applies discourse analysis to get the pairwise discourse relation set denoted as $\mathcal{R}_{sep} \subseteq \mathcal{V} \times \mathcal{V}$.

Step 2: Discourse Graph. The discourse graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ consists of the sentences nodes and discourse relations discussed above. Given the document doc and the discourse relation set $\mathcal{R}_{dis}$, for each relation $(sent_i, sent_j) \in \mathcal{R}_{dis}$, one adds the nodes v_i (for the sentence $sent_i$) and v_j (for the sentence $sent_j$) and add a directed edge from node v_i to node v_j.

Knowledge Graph Construction Knowledge Graph (KG) that captures entities and relations can greatly facilitate learning and reasoning in many

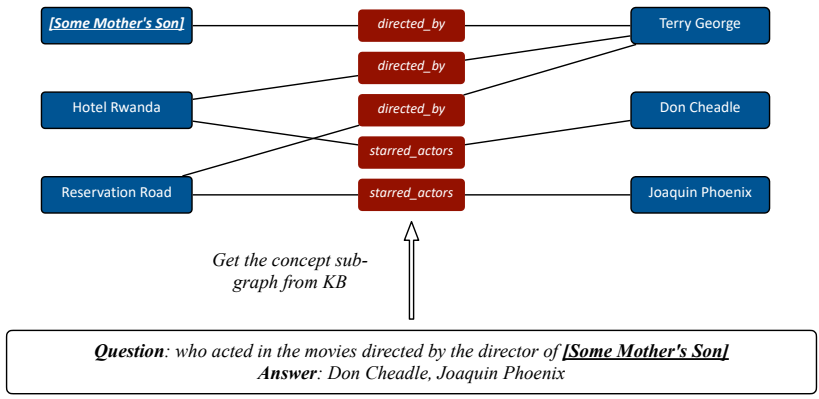


Figure 5: An example for knowledge graph construction, where the knowledge base (KB) used and the generated concept graph are both from the dataset *MetaQA* (Zhang *et al.*, 2018d).

NLP applications. In general, the KGs can be divided into two main categories depending on their graph construction approaches. Many applications treat the KG as the compact and interpretable intermediate representation of the unstructured data (e.g., the document) (Wu *et al.*, 2020c; Koncel-Kedziorski *et al.*, 2019; Huang *et al.*, 2020b). Conceptually, it is almost similar to the IE graph, which we have discussed previously. On the other hand, many other works (Wu *et al.*, 2020b; Ye *et al.*, 2019; Bansal *et al.*, 2019; Yang *et al.*, 2019) incorporate the existing knowledge bases such as YAGO (Suchanek *et al.*, 2008)) and ConceptNet (Speer *et al.*, 2017) to further enhance the performance of downstream tasks (Zhao *et al.*, 2020c). In what follows, we will briefly discuss the second category of KG from the view of the graph construction.

The KG can be denoted as $\mathcal{G}(\mathcal{V}, \mathcal{E})$, which is usually constructed by elements in knowledge base. Formally, one defines the triple (e_1, rel, e_2) as the basic elements in the knowledge base, in which e_1 is the source entity, e_2 is the target entity, and rel is the relation type. Then one adds two nodes v_1 (for the source element e_1) and v_2 (for the target element e_2) in the KG and add a directed edge from node v_1 to node v_2 with edge type rel. An example of such KG is shown in Fig. 5.

It is worth noting that the KG plays different roles in various applications. In some applications (e.g. knowledge graph completion and knowledge base question answering), KG is always treated as part of the inputs. In this

scenario (Ye *et al.*, 2019; Zhang *et al.*, 2020g; Li *et al.*, 2019; Wu *et al.*, 2020a), researchers typically use the whole KG $\mathcal{G}$ as the learning object. But for some other applications (e.g. natural language translation), the KG can be treated as the data augmentation method. In this case, the whole KG such as ConceptNet (Speer *et al.*, 2017) is usually too large and noisy for some domain-specific applications (Kapanipathi *et al.*, 2020; Lin *et al.*, 2019a), and thus it is not suitable to use the whole graph as inputs. In contrast, as shown in Figure 5, one instead usually constructs subgraphs from the given query (it is often the text-based inputs like the queries in the reading comprehension task) (Xu *et al.*, 2019b; Teru *et al.*, 2020; Kapanipathi *et al.*, 2020).

The construction methods could may vary dramatically in the literature. Here, we only present one representative method for illustration purpose (Teru *et al.*, 2020). The first thing for constructing KG is to fetch the term instances in the given query. Then, they could link the term instances to the concepts in the KG by some matching algorithms such as max-substring matching. The concepts are regarded as the initial nodes in the extracted subgraph. Next step is to fetch the 1-hop neighbors of the initial nodes in the KG. Additionally, one may calculate the relevance of the neighbors with the initial nodes by applying some graph node relevance model such as the Personalized PageRank (PPR) algorithm (Page *et al.*, 1999). Then based on the results, one may further prune out the edges with relevance score that is below the confidence threshold and remove the isolated neighbors. The remaining final subgraph is then used to feed any graph representation learning module later.

Coreference Graph Construction In linguistics, coreference (or co-reference) occurs when two or more terms in a given paragraph refer to the same object. Many works have demonstrated that such phenomenon is helpful for better understanding the complex structure and logic of the corpus and resolve the ambiguities (Xu *et al.*, 2020c; De Cao *et al.*, 2018; Sahu *et al.*, 2019). To effectively leverage the coreference information, the coreference graph is constructed to explicitly model the implicit coreference relations. Given a set of phrases, a coreference graph can link the nodes (phrases) which refer to the same entity in the text corpus. In the following subsection, we focus on the coreference graph construction for a paragraph $para$ consisting of m sentences. We will briefly discuss the coreference relation and then discuss the approaches for building the coreference graph in various NLP tasks (De Cao

et al., 2018; Sahu *et al.*, 2019; Qian *et al.*, 2019; Xu *et al.*, 2020c; Xu *et al.*, 2020a; Luan *et al.*, 2019). It is worth noting that although it is similar to the IE graph's first step, the coreference graph will explicitly model the coreference relationship by graph instead of collapse into one node.

Step 1: Coreference Relation. The coreference relations can be obtained easily by the coreference resolution system, as discussed in IE graph construction. Similarly, we can obtain the coreference clusters $\mathcal{C}$ given a specific paragraph. All phrases in a cluster $C_i \in \mathcal{C}$ refer to the same object.

Step 2: Coreference Graph. The coreference graph are built on the coreference relation set $\mathcal{R}_{coref}$. It can be generally divided into two main category depending on the node type: 1) phrases (or mentions) (Koncel-Kedziorski *et al.*, 2019; Luan *et al.*, 2019; De Cao *et al.*, 2018), 2) words (Sahu *et al.*, 2019). For the first class, the coreference graph $\mathcal{G}$ consists of all mentions in relation set $\mathcal{R}_{coref}$. For each phrase pair p_i, p_j in cluster $C_k \in \mathcal{C}$, one may add an undirected edge between node v_i (for the phrase p_i) and node v_j (for the phrase p_j). For the second case, the coreference graph $\mathcal{G}$ consists of words. One minor difference is that one only links the first word of each phrase for each associated phrases.

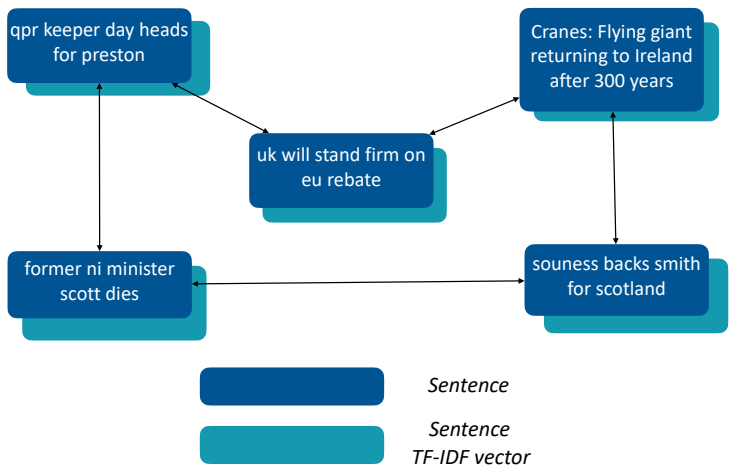


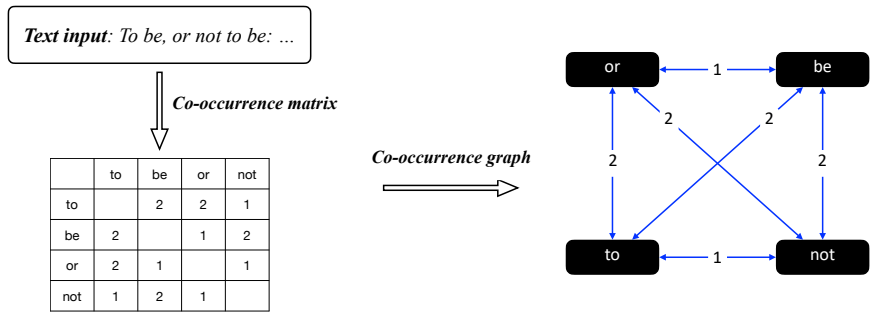
Figure 6: An example for similarity graph construction. We use sentences as nodes and initialize their features with *TF-IDF* vectors.

Similarity Graph Construction The similarity graphs aim to quantify the similarity between nodes, which are widely used in many NLP tasks (Liu *et al.*, 2019a; Linmei *et al.*, 2019; Yasunaga *et al.*, 2017). Since the similarity graph is typically application-oriented, we focus on the basic procedure of constructing the similarity graph for various types of elements such as entities, sentences and documents, and neglect the application specific details. It is worth noting that the similarity graph construction is conducted during preprocessing and is not jointly trained with the remaining learning system in an end-to-end manner. One example of similarity graph is shown in Fig. 6.

Step 1: Similarity Graph. Given a corpus C, in a similarity graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, the graph nodes can be defined in different granularity levels such as entities, sentences and documents. We denote the basic node set as $\mathcal{V}$ regardless of specific node types. One can calculate the node features by various mechanisms such as TF-IDF for sentences (or documents) (Liu *et al.*, 2019a; Yasunaga *et al.*, 2017) and embeddings for entities (Linmei *et al.*, 2019). Then, similarity scores between node pairs can be computed by various metrics such as cosine similarity (Liu *et al.*, 2019a; Linmei *et al.*, 2019; Yasunaga *et al.*, 2017), and used to indicate edge weights of the node pairs.

Step 2: Sparse mechanism. The initial similarity graph is typically dense even some edge weights are very small or even negative. These values can be treated as noise, which plays little roles in the similarity graph. Thus various sparse techniques are proposed to further improve the quality of graph by sparsifying a graph. One widely used sparse method is k-NN (Liu *et al.*, 2019a). Specifically, for node v_i and its' neighbor set $N(v_i)$, one only reserves edges by keeping k largest edge weights and dropping the remaining edges. The other widely used method is $\epsilon-$sparse (Linmei *et al.*, 2019; Yasunaga *et al.*, 2017). In particular, one will remove the edges whose weights are smaller than the certain threshold ϵ.

Co-occurrence Graph Construction The co-occurrence graph aims to capture the co-occurrence relation between words in the text, which is widely used in many NLP tasks (Christopoulou *et al.*, 2019; Zhang and Qian, 2020; Zhang *et al.*, 2020f). The co-occurrence relation, which describes the frequency of two words that co-occur within a fix-sized context window, is an important feature capturing the semantic relationship among words in the corpus. In what follows, we will first present the approaches of obtaining the



	to	be	or	not
to		2	2	1
be	2		1	2
or	2	1		1
not	1	2	1	

Figure 7: An example for co-occurrence graph construction where edge weights stand for the co-occurrence frequency between words. We set the window size as 3.

co-occurrence relations and then discuss the basic procedure of building a co-occurrence graph for a corpus C. An example of co-occurrence graph can be seen in Fig. 7.

Step 1: Co-occurrence Relation. The co-occurrence relation is defined by the co-occurrence matrix of the given corpus C. For a specific paragraph $para$ consists of m sentences, the co-occurrence matrix describes how words occur together. One may denote the co-occurrence the matrix as $\mathbf{M} \in \mathbb{R}^{|V|\times|V|}$, where $|V|$ is the vocabulary size of C. $\mathbf{M}_{w_i,w_j}$ describes how many times word w_i, w_j occur together within a fix-size sliding windows in the corpus C. After obtaining the co-occurrence matrix, there are two typical methods to calculate the weights between words: 1) co-occurrence frequency (Zhang *et al.*, 2020f; Christopoulou *et al.*, 2019; Zhang and Qian, 2020; Edouard *et al.*, 2017; Zhu *et al.*, 2018) and 2) point-wise mutual information (PMI) (Yao *et al.*, 2019b; Hu *et al.*, 2020b; Hu *et al.*, 2019b).

Step 2: Co-occurrence Graph. The co-occurrence graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ consists of the words nodes and co-occurrence relations discussed above. Given the corpus C and the co-occurrence relation set $\mathcal{R}_{co}$, for each relation $(w_i, w_j) \in \mathcal{R}_{co}$, one adds the nodes v_i (for the word w_i) and v_j (for the word w_j) and add an undirected edge from node v_i to node v_j initialized with the aforementioned calculated edge weights.

Topic Graph Construction The topic graph is built on several documents, which aims to model the high-level semantic relations among different topics (Linmei *et al.*, 2019; Li *et al.*, 2020b). In particular, given a set of docu-

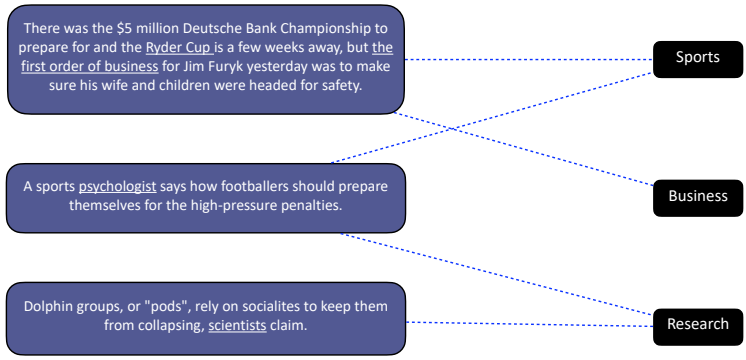


Figure 8: An example for topic graph construction, where the dash line stands for the topic modeling process by leveraging the LDA algorithm on dataset *AG news* (Zhang *et al.*, 2015).

ments $\mathcal{D} = \{doc_1, doc_2, ..., doc_m\}$, one first learns the latent topics denoted as $\mathcal{T}$ using some topic modeling algorithms such as LDA (Blei *et al.*, 2003). Then one could construct the topic graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with $\mathcal{V} = \mathcal{D} \cup \mathcal{T}$. The undirected edge between the node v_i (for a document) and the node v_j (for a topic) is built only if the document has that topic. An example of topic graph is shown in Fig. 8.

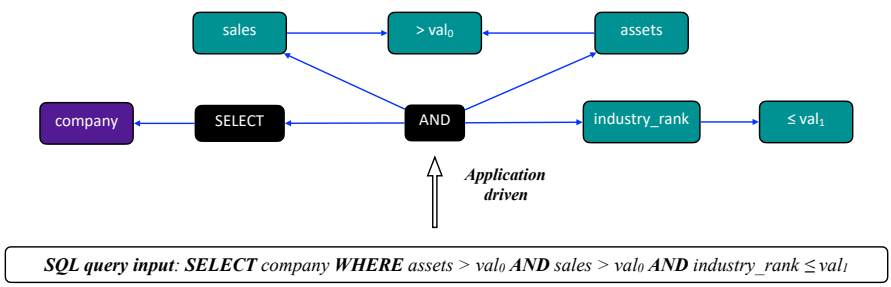


Figure 9: An example for application-driven graph construction, which is specially designed for SQL query input.

App-driven Graph Construction The app-driven graphs refer to the graph specially designed for specific NLP tasks (Gui *et al.*, 2019; Ding *et al.*, 2019b; Yin *et al.*, 2020; Luo and Zhao, 2020), which cannot be trivially covered by the previously discussed static graph types. In some NLP tasks,

it is common to represent unstructured data by structured formation with application-specific approaches. For example, the SQL language can be naturally represented by the SQL parsing tree. Thus it can be converted to the SQL graph (Xu *et al.*, 2018a; Bogin *et al.*, 2019a; Huo *et al.*, 2019; Bogin *et al.*, 2019b). Since these graphs are too specialized based on the domain knowledge, there are no unified pattern to summarize how to build an app-driven graph. An example of such application-driven graph like SQL graph is illustrated in Fig. 9. In Sec. 0.7, we will further discuss how these graph construction methods are used in various popular NLP tasks.

Hybrid Graph Construction and Discussion

Most previous static graph construction methods only consider one specific relation between nodes. Although the obtained graphs capture the structure information well to some extent, they are also limited in exploiting different types of graph relations. To address this limitation, there is an increasing interest in building a hybrid graph by combing several graphs together in order to enrich the semantic information in graph (Jia *et al.*, 2020; Sahu *et al.*, 2019; Xu *et al.*, 2018c; Zeng *et al.*, 2020; Xu *et al.*, 2020c; Xu *et al.*, 2020a; Christopoulou *et al.*, 2019; Yao *et al.*, 2019b). The method of constructing a hybrid graph is mostly application specific, and thus we only present some representative approach for such a graph construction.

To capture multiple relations, a common strategy is to construct a heterogeneous graph, which contains multiple types of nodes and edges. Without losing generality, we assume that one may create a hybrid graph $\mathcal{G}_{hybrid}$ with two different graph sources $\mathcal{G}_a(\mathcal{V}_a, \mathcal{E}_a)$ and $\mathcal{G}_b(\mathcal{V}_b, \mathcal{E}_b)$. Graphs a, b are two different graph types such as *dependency graph* and *constituency graph*. Given these textual inputs, if $\mathcal{G}_a$ and $\mathcal{G}_b$ share the same node sets (i.e., $\mathcal{V}_a = \mathcal{V}_b$), we merge the edge sets by annotating relation-specific edge types (Xu *et al.*, 2018c; Sahu *et al.*, 2019; Zeng *et al.*, 2020; Xu *et al.*, 2020c; Xu *et al.*, 2020a). Otherwise, we merge the $\mathcal{V}_a$ and $\mathcal{V}_b$ to get the hybrid node set, denoted as $\mathcal{V} = \mathcal{V}_a \cup \mathcal{V}_b$ (Jia *et al.*, 2020; Christopoulou *et al.*, 2019). Then we generate $\mathcal{E}_a$ and $\mathcal{E}_b$ to $\mathcal{E}$ by mapping the source and target nodes from $\mathcal{V}_a$ and $\mathcal{V}_b$ to $\mathcal{V}$.

0.4.2 Dynamic Graph Construction

Although static graph construction has the advantage of encoding prior knowledge of the data into the graph structure, it has several limitations. First of all, extensive human efforts and domain expertise are needed in order to construct a reasonably performant graph topology. Secondly, the manually constructed graph structure might be error-prone (e.g., noisy or incomplete). Thirdly, since the graph construction stage and graph representation learning stage are disjoint, the errors introduced in the graph construction stage cannot be corrected and might be accumulated to later stages, which can result in degraded performance. Lastly, the graph construction process is often informed solely by the insights of the NLP practitioners, and might be sub-optimal for the downstream prediction task.

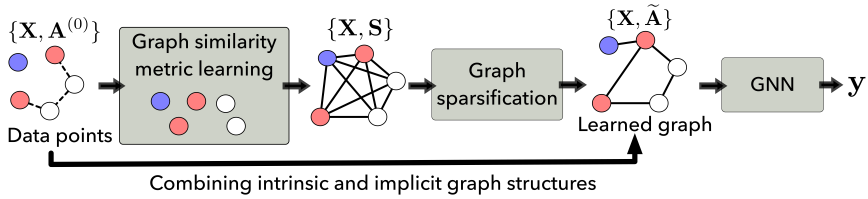


Figure 10: Overall illustration of dynamic graph construction approaches. Dashed lines (in data points on left) indicate the optional intrinsic graph topology.

In order to tackle the above challenges, recent attempts on GNN for NLP (Chen *et al.*, 2020f; Chen *et al.*, 2020i; Liu *et al.*, 2021b; Liu *et al.*, 2019c) have explored dynamic graph construction without resorting to human efforts or domain expertise. Most dynamic graph construction approaches aim to dynamically learn the graph structure (i.e., a weighted adjacency matrix) on the fly, and the graph construction module can be jointly optimized with the subsequent graph representation learning modules toward the downstream task in an end-to-end manner. One good example of dynamic graph construction is when constructing a graph capturing the semantic relationships among all the words in a text passage in the task of conversational machine reading comprehension (Reddy *et al.*, 2019), instead of building a fixed static graph based on domain expertise, one can jointly train a graph structure learning module together with the graph embedding learning module in order to learn an optimal graph structure considering not only the semantic meanings of the words but also the conversation history and current question.

As shown in Fig. 10, these dynamic graph construction approaches typi-

cally consist of a graph similarity metric learning component for learning an adjacency matrix by considering pair-wise node similarity in the embedding space, and a graph sparsification component for extracting a sparse graph from the learned fully-connected graph. It is reported to be beneficial to combine the intrinsic graph structures and learned implicit graph structures for better learning performance (Li *et al.*, 2018a; Chen *et al.*, 2020f; Liu *et al.*, 2021b). Moreover, in order to effectively conduct the joint graph structure and representation learning, various learning paradigms have been proposed. In what follows, we will discuss all these effective techniques for dynamic graph construction. More broadly speaking, graph structure learning for GNNs itself is a trending research problem in the machine learning field, and has been actively studied beyond the NLP community (Li *et al.*, 2018a; Norcliffe-Brown *et al.*, 2018; Velickovic *et al.*, 2020; Kalofolias and Perraudin, 2019; Franceschi *et al.*, n.d.). However, in this survey, we will focus on its recent advances in the NLP field, as shown in Table 2. We hereafter use dynamic graph construction and graph structure learning interchangeably.

Graph Similarity Metric Learning Techniques

Based on the assumption that node attributes contain useful information for learning the implicit graph structure, recent work has explored to cast the graph structure learning problem as the problem of similarity metric learning defined upon the node embedding space. The learned similarity metric function can be later applied to an unseen set of node embeddings to infer a graph structure, thus enabling inductive graph structure learning. For data deployed in non-Euclidean domains like graphs, the Euclidean distance is not necessarily the best metric for measuring node similarity. Various similarity metric functions have been proposed for graph structure learning for GNNs. According to the types of information sources utilized, we group these metric functions into two categories: *Node Embedding Based Similarity Metric Learning* and *Structure-aware Similarity Metric Learning*.

Node Embedding Based Similarity Metric Learning Node embedding based similarity metric functions are designed to learn a weighted adjacency matrix by computing the pair-wise node similarity in the embedding space. Common metric functions include attention-based metric functions and

cosine-based metric functions.

Attention-based Similarity Metric Functions. Most of the similarity metric functions proposed so far are based on the attention mechanism (Bahdanau *et al.*, 2015; Vaswani *et al.*, 2017). In order to increase the learning capacity of dot product based attention mechanism, Chen *et al.* (2020h) proposed a modified dot product by introducing learnable parameters, formulated as follows:

$$S_{i,j} = (\mathbf{v}_i \odot \mathbf{u})^T \mathbf{v}_j \tag{19}$$

where $\mathbf{u}$ is a non-negative weight vector learning to highlight different dimensions of the node embeddings, and $\odot$ denotes element-wise multiplication.

Similarly, Chen *et al.* (2020i) designed a more expressive version of dot product by introducing a learnable weight matrix, formulated as follows:

$$S_{i,j} = \mathrm{ReLU}(\mathbf{W}\mathbf{v}_i)^T \mathrm{ReLU}(\mathbf{W}\mathbf{v}_j) \tag{20}$$

where $\mathbf{W}$ is a $d \times d$ weight matrix, and $\mathrm{ReLU}(x) = \max(0, x)$ is a rectified linear unit (ReLU) (Nair and Hinton, 2010) used to enforce the sparsity of the similarity matrix.

Cosine-based Similarity Metric Functions. Chen *et al.* (2020f) extended the vanilla cosine similarity to a multi-head weighted cosine similarity to capture pair-wise node similarity from multiple perspectives, formulated as follows:

$$\begin{aligned} S_{i,j}^p &= \cos(\mathbf{w}_p \odot \mathbf{v}_i, \mathbf{w}_p \odot \mathbf{v}_j) \\ S_{i,j} &= \frac{1}{m} \sum_{p=1}^{m} S_{ij}^p \end{aligned} \tag{21}$$

where $\mathbf{w}_p$ is a weight vector associated to the p-th perspective, and has the same dimension as the node embeddings. Intuitively, $S_{i,j}^p$ computes the pair-wise cosine similarity for the p-th perspective where each perspective considers one part of the semantics captured in the embeddings. Besides increasing the learning capacity, employing multi-head learners is able to stabilize the learning process, which has also been observed in (Vaswani *et al.*, 2017; Velickovic *et al.*, 2018).

Structure-aware Similarity Metric Learning Inspired by structure-aware transformers (Zhu *et al.*, 2019c; Cai and Lam, 2020b), recent approaches employ structure-aware similarity metric functions that additionally consider

the existing edge information of the intrinsic graph beyond the node information. For instance, Liu *et al.* (2019c) proposed a structure-aware attention mechanism for learning pair-wise node similarity, formulated as follows:

$$S_{i,j}^{l} = \text{softmax}(\mathbf{u}^T \tanh(\mathbf{W}[\mathbf{h}_i^l, \mathbf{h}_j^l, \mathbf{v}_i, \mathbf{v}_j, \mathbf{e}_{i,j}])) \tag{22}$$

where $\mathbf{v}_i$ represents the embedding of node i, $\mathbf{e}_{i,j}$ represents the embedding of the edge connecting node i and j, $\mathbf{h}_i^l$ is the embedding of node i in the l-th GNN layer, and $\mathbf{u}$ and $\mathbf{W}$ are trainable weight vector and weight matrix, respectively.

Similarly, Liu *et al.* (2021b) introduced a structure-aware global attention mechanism, formulated as follows,

$$S_{i,j} = \frac{\text{ReLU}(\mathbf{W}^Q \mathbf{v}_i)^T (\text{ReLU}(\mathbf{W}^K \mathbf{v}_i) + \text{ReLU}(\mathbf{W}^R \mathbf{e}_{i,j}))}{\sqrt{d}} \tag{23}$$

where $\mathbf{e}_{i,j}$ is the embedding of the edge connecting node i and j, and $\mathbf{W}^Q$, $\mathbf{W}^K$, and $\mathbf{W}^R$ are linear transformations that map the node and edge embeddings to the latent embeddding space.

Graph Sparsification Techniques

Most graphs in real-world scenarios are sparse graphs. Similarity metric functions consider relations between any pair of nodes and returns a fully-connected graph, which is not only computationally expensive but also might introduce noise such as unimportant edges. Therefore, it can be beneficial to explicitly enforce sparsity to the learned graph structure. Besides applying the ReLU function in the similarity metric functions (Chen *et al.*, 2020i; Liu *et al.*, 2021b), various graph sparsification techniques have been adopted to enhance the sparsity of the learned graph structure.

Chen *et al.* (2020i) and Chen *et al.* (2020h) applied a kNN-style sparsification operation to obtain a sparse adjacency matrix from the node similarity matrix computed by the similarity metric learning function, formulated as follows:

$$\mathbf{A}_{i,:} = \text{topk}(\mathbf{S}_{i,:}) \tag{24}$$

where for each node, only the K nearest neighbors (including itself) and the associated similarity scores are kept, and the remaining similarity scores are masked off.

Chen *et al.* (2020f) enforced a sparse adjacency matrix by considering only the ε-neighborhood for each node, formulated as follows:

$$A_{i,j} = \begin{cases} S_{i,j} & S_{i,j} > \varepsilon \\ 0 & \text{otherwise} \end{cases} \tag{25}$$

where those elements in S which are smaller than a non-negative threshold ε are all masked off (i.e., set to zero).

Besides explicitly enforcing the sparsity of the learned graph by applying certain form of threshold, sparsity has also been enforced implicitly in a learning-based manner. Chen *et al.* (2020f) introduced the following regularization term to encourage sparse graphs.

$$\frac{1}{n^2}||A||_F^2 \tag{26}$$

where $||\cdot||_F$ denotes the Frobenius norm of a matrix.

Combining Intrinsic Graph Structures and Implicit Graph Structures

Recent studies (Li *et al.*, 2018a; Chen *et al.*, 2020f; Liu *et al.*, 2021b) have shown that it could hurt the downstream task performance if the intrinsic graph structure is totally discard while doing dynamic graph construction. This is probably because the intrinsic graph typically still carries rich and useful information regarding the optimal graph structure for the downstream task. They therefore proposed to combine the learned implicit graph structure with the intrinsic graph structure based on the assumption that the learned implicit graph is potentially a "shift" (e.g., substructures) from the intrinsic graph structure which is supplementary to the intrinsic graph structure. The other potential benefit is incorporating the intrinsic graph structure might help accelerate the training process and increase the training stability. Since there is no prior knowledge on the similarity metric and the trainable parameters are randomly initialized, it may usually take long time to converge.

Different ways for combining intrinsic and implicit graph structures have been explored. For instance, Li *et al.* (2018a) and Chen *et al.* (2020f) proposed to compute a linear combination of the normalized graph Laplacian of the intrinsic graph structure $L^{(0)}$ and the normalized adjacency matrix of the

implicit graph structure $\mathrm{f}(A)$, formulated as follows:

$$\widetilde{A} = \lambda L^{(0)} + (1 - \lambda)\mathrm{f}(A) \tag{27}$$

where $\mathrm{f} : \mathbb{R}^{n \times n} \rightarrow \mathbb{R}^{n \times n}$ can be arbitrary normalization operations such as graph Laplacian operation (Li *et al.*, 2018a) and row-normalization operation (Chen *et al.*, 2020f). Instead of explicitly fusing the two graph adjacency matrices, Liu *et al.* (2021b) proposed a hybrid message passing mechanism for GNNs which fuses the two aggregated node vectors computed from the intrinsic graph and the learned implicit graph, respectively, and then feed the fused vector to a GRU to update node embeddings.

Learning Paradigms

Most existing dynamic graph construction approaches for GNNs consist of two key learning components: graph structure learning (i.e., similarity metric learning) and graph representation learning (i.e., GNN module), and the ultimate goal is to learn the optimized graph structures and representations with respect to certain downstream prediction task. How to optimize the two separate learning components toward the same ultimate goal becomes an important question. Here we highlight three representative learning paradigms. The most straightforward strategy (Chen *et al.*, 2020i; Chen *et al.*, 2020h; Liu *et al.*, 2021b) is to jointly optimize the whole learning system in an end-to-end manner toward the downstream (semi-)supervised prediction task. Another common strategy (Yang *et al.*, 2018b; Liu *et al.*, 2019c; Huang *et al.*, 2020a) is to adaptively learn the input graph structure to each stacked GNN layer to reflect the changes of the intermediate graph representations. This is similar to how transformer models learn different weighted fully-connected graphs in each layer. Unlike the above two paradigms, Chen *et al.* (2020f) proposed an iterative graph learning framework by learning a better graph structure based on better graph representations, and in the meantime, learning better graph representations based on a better graph structure in an iterative manner. As a result, this iterative learning paradigm repeatedly refines the graph structure and the graph representations toward the optimal downstream performance.

0.5 Graph Representation Learning for NLP

In the previous section, we have presented various graph construction methods, including static graph construction and dynamic graph construction. In this section, we will discuss various graph representation learning techniques that are directly operated on the constructed graphs for various NLP tasks. The goal of graph representation learning is to find a way to incorporate information of graph structures and attributes into a low-dimension embeddings via a machine learning model (Hamilton *et al.*, 2017b). To mathematically formalize this problem, we give the mathematical notations for arbitrary graphs as $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{R})$, where $\mathcal{V}$ is the node set, $\mathcal{E}$ is the edge set, $\mathcal{T} = \{T_1, T_2, ..., T_p\}$ is the collection of node types, and $\mathcal{R} = \{R_1, ..., R_q\}$ is the collection of edge types. $|\cdot|$ is the number of elements. $\tau(\cdot) \in \mathcal{T}$ is the node type indicator function (e.g., $\tau(v_i) \in \mathcal{T}$ is the type of node v_i), and $\phi(\cdot) \in \mathcal{R}$ is the edge type indicator function (e.g., $\phi(e_{i,j}) \in \mathcal{R}$ is the type of edge $e_{i,j}$), respectively.

Generally speaking, the constructed graphs from the raw text data are either homogeneous or heterogeneous graphs. Thus, in Section 0.5.1, we will discuss various graph representation learning methods for homogeneous graphs, including scenarios for the original homogeneous graph and some converting from heterogeneous graphs. In Section 0.5.2, we will discuss the GNN-based methods for multi-relational graphs, and in Section 0.5.3, we will discuss the GNNs for dealing with the heterogeneous graphs.

0.5.1 GNNs for Homogeneous Graphs

By definition, a graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{R}), s.t. |\mathcal{T}| = 1, |\mathcal{R}| = 1$ is called homogeneous graph. Most graph neural networks such as GCN, GAT, and GraphSage are designed for homogeneous graphs, which, however, can not fit well in many NLP tasks. For example, given a natural language text, the constructed dependency graph is arbitrary graph that contains multiple relations, which cannot be exploited by traditional GNN methods directly. Thus, in this subsection, we will first discuss the various strategies for converting arbitrary graphs to homogeneous graphs, including static graphs and dynamic graphs. Then, we will discuss the graph neural networks considering bidirectional encoding.

Static Graph: Treating Edge Information as Connectivity

GNNs for dealing with the static graphs normally consists of two stages, namely, converting edge information and node representation learning, as described in the following.

Converting Edge Information to Adjacent Matrix Basically, the edges are viewed as the connection information between nodes. In this case, it is normal to discard the edge type information and retain the connections to convert the heterogeneous graphs (Yang *et al.*, 2019; Zhang *et al.*, 2019a; Yao *et al.*, 2019b; Wu *et al.*, 2019b; Li *et al.*, 2019) to homogeneous graphs. After obtaining such a graph, we can represent the topology of the graph as a unified adjacency matrix A. Specifically, for an edge $e_{i,j} \in \mathcal{E}$ which connect node v_i and v_j, $A_{i,j}$ denotes to the edge weight for weighted static graph, or $A_{i,j} = 1$ for unweighted connections and $A_{i,j} = 0$ otherwise. The static graphs can also be divided into directed and undirected graphs. For the undirected case, the adjacency matrix is symmetric matrix, which means $A_{i,j} = A_{j,i}$. And for the other case, it is always not symmetric. The $A_{i,j}$ is strictly defined by the edge from node v_i to node v_j. It is worth noting that the directed graphs can be transformed to undirected graphs (Yasunaga *et al.*, 2017) by averaging the edge weights in both directions. The edge weights are rescaled, whose maximum edge weight is 1 before feeding to the GNN.

Node Representation Learning Next, given initial node embedding $\mathbf{X}$ and adjacency matrix A, the node representation is extracted base on the classical GNNs techniques. For undirected graphs, most works (Liu *et al.*, 2019b; Wang *et al.*, 2018; Yao *et al.*, 2019b; Zhang *et al.*, 2020e) mainly adopt graph representation learning algorithms such as GCN, GGNN, GAT, GraphSage, etc. and stack them to explore deep semantic correlations in the graph. When it comes to the directed graphs, few GNN methods such as GGNN, GAT still work (Chen *et al.*, 2020h; Wang *et al.*, 2020d; Qiu *et al.*, 2019; Yan *et al.*, 2019; Ji *et al.*, 2019; Sui *et al.*, 2019). While for the other GNNs that can not be directly applied into directed graphs, the simple strategy is to ignore the directions (i.e., converting the directed graphs to undirected graphs) (Yasunaga *et al.*, 2017; Wang *et al.*, 2018; Liu *et al.*, 2019b). However, such methods allow the message to propagate in both

directions without constraints. To solve this problem, many efforts have been made to adapt the GNN to directed graphs. For GCN, some spatial-based GCN algorithms are designed for directed graphs such as DCNN (Atwood and Towsley, 2016). GraphSage can be easily extended to directed graphs by modifying the aggregation function via specifying the edge directions and aggregating them separately) (Xu *et al.*, 2018b).

Dynamic Graph

Dynamic graphs that aim to learn the graph structure together with the downstream task jointly are widely adopted by graph representation learning (Chen *et al.*, 2020i; Chen *et al.*, 2020h; Chen *et al.*, 2020f; Hashimoto and Tsuruoka, 2017; Guo *et al.*, 2019b). Early works mainly adopt the recurrent network by treating the graph node embeddings as RNN's state encodings (Hashimoto and Tsuruoka, 2017), which can be regarded as the rudiment of GGNN. Then the classic GNNs such as GCN (Guo *et al.*, 2019b), GAT (Cui *et al.*, 2020b), GGNN (Chen *et al.*, 2020i; Chen *et al.*, 2020h) are utilized to learn the graph embedding effectively. Recent researchers adopt attention-based or metric learning-based mechanisms to learn the implicit graph structure (i.e., the graph adjacency matrix A) from unstructured texts. The learning process of graph structure is jointly with the downstream tasks via an end-to-end fashion (Shaw *et al.*, 2018; Chen *et al.*, 2020h; Luan *et al.*, 2019; Chen *et al.*, 2020i).

Graph Neural Networks: Bidirectional Graph Embeddings

In the previous sub-sections, we present the typical techniques for constructing and learning node embeddings from the static homogeneous graphs. In this subsection, we provide a detailed discuss on how to handle the edge directions. In reality, many graphs are directed acyclic graphs (DAG) (Cai and Lam, 2020b), which information is propagated along the specific edge direction. However, some researchers allow the information propagate equally in both directions (Yasunaga *et al.*, 2017; Wang *et al.*, 2018; Liu *et al.*, 2019b) and others discard the information containing in outgoing edges (Yan *et al.*, 2019; Wang *et al.*, 2020d; Qiu *et al.*, 2019; Ji *et al.*, 2019; Sui *et al.*, 2019), both of which will lose some important structure information for the final representation learning.

To deal with this, bidirectional graph neural network (bidirectional GNN) is proposed to learn the node representation from both incoming and outgoing edges in a interleaved fashion. To introduce different variants of bidirectional GNN, we first give some unified mathematical notations. For a specific node $v_i \in \mathcal{V}$ in the graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ and its neighbor nodes $N(v_i)$ (i.e. any node v_j satisfy $e_{i,j} \in \mathcal{E}$ or $e_{j,i} \in \mathcal{E}$), we define the incoming (backward) nodes set as $N_{\dashv}(v_i)$ satisfying $e_{j,i} \in \mathcal{E}, v_j \in N_{\dashv}(v_i)$ and outgoing (forward) nodes set as $N_{\vdash}(v_i)$ holding $e_{i,j} \in \mathcal{E}, v_j \in N_{\vdash}(v_i)$.

Xu *et al.* (2018b) firstly extend the GraphSage to a bi-directional version by calculating the graph embedding separately for each direction and combine them at last. At each computation hop, for each node in the graph, they aggregate the incoming nodes and outgoing nodes separately to get backward and forward immediate-aggregated representation as follows:

$$\begin{aligned} \mathbf{h}_{i,\dashv}^{(k)} &= \sigma(\mathbf{W}^{(k)} \cdot f_k^{\dashv}(\mathbf{h}_{i,\dashv}^{(k-1)}, \{\mathbf{h}_{j,\dashv}^{(k-1)}, \forall v_j \in N_{\dashv}(v_i)\})), \\ \mathbf{h}_{i,\vdash}^{(k)} &= \sigma(\mathbf{W}^{(k)} \cdot f_k^{\vdash}(\mathbf{h}_{i,\vdash}^{(k-1)}, \{\mathbf{h}_{j,\vdash}^{(k-1)}, \forall v_j \in N_{\vdash}(v_i)\})), \end{aligned} \tag{28}$$

where $k \in \{1, 2, ..., K\}$ denotes the layer number, and $\mathbf{h}_{i,\dashv}^{(k)}, \mathbf{h}_{i,\vdash}^{(k)}$ denote the backward and forward aggregated results respectively. At the final step, the final forward and backward representation is concatenated to calculate the final bi-directional representation.

Although works effectively, the bidirectional GraphSage learns both directions separately. To this end, Chen *et al.* (2020i) proposes the bidirectional GGNN to address this issue. Technically, at each iteration, after obtaining aggregated vector representations $\mathbf{h}_{i,\vdash}^{(k)}, \mathbf{h}_{i,\dashv}^{(k)}$, they opt to fuse them into one vector as follows:

$$\mathbf{h}_{N(v_i)}^{(k)} = \text{Fuse}(\mathbf{h}_{i,\vdash}^{(k)}, \mathbf{h}_{i,\dashv}^{(k)}), \tag{29}$$

where the function $\text{Fuse}(\cdot, \cdot)$ is the gated sum of two information sources:

$$\text{Fuse}(\mathbf{a}, \mathbf{b}) = \mathbf{z} \odot \mathbf{a} + (1 - \mathbf{z}) \odot \mathbf{b}, \mathbf{z} = \sigma(\mathbf{W}_z[\mathbf{a}, \mathbf{b}, \mathbf{a} \odot \mathbf{b}, \mathbf{a} - \mathbf{b}] + \mathbf{b}_z) \tag{30}$$

where $\mathbf{a} \in \mathbb{R}^d, \mathbf{b} \in \mathbb{R}^d$ are inputs, $\mathbf{W}_z \in \mathbb{R}^{d \times 4d}, \mathbf{b}_z \in \mathbb{R}^d$ are learnable weights and $\sigma(\cdot)$ is the sigmoid function. Then, a Gated Recurrent Unit (GRU) is used to update the node embeddings by incorporating the aggregation information as follows:

$$\mathbf{h}_i^{(k)} = \text{GRU}(\mathbf{h}_i^{(k-1)}, \mathbf{h}_{N(v_i)}^{(k)}). \tag{31}$$

Unlike previous methods, which are specially designed for the specific GNN methods, Ribeiro *et al.* (2019a) further proposes a general bidirectional GNN framework, which can be easily applied to most existing GNNs. Technically, they first encode the graph in two directions:

$$\begin{aligned}\mathbf{h}_{i,\dashv}^{(k)} &= \text{GNN}(\mathbf{h}_i^{(k-1)}, \{\mathbf{h}_j^{(k-1)} : \forall v_j \in N_{\dashv}(v_i)\}), \\ \mathbf{h}_{i,\vdash}^{(k)} &= \text{GNN}(\mathbf{h}_i^{(k-1)}, \{\mathbf{h}_j^{(k-1)} : \forall v_j \in N_{\vdash}(v_i)\}),\end{aligned} \tag{32}$$

where $\text{GNN}(\cdot)$ can denote to any variant of GNNs. Similar to strategy in the bidirectional RNNs (Schuster and Paliwal, 1997), they learn the forward and backward directions separately and concatenate them together with the original node feature as follows:

$$\mathbf{h}_i^{(k)} = [\mathbf{h}_{i,\dashv}^{(k)}, \mathbf{h}_{i,\vdash}^{(k)}, \mathbf{h}_i^{(k-1)}]. \tag{33}$$

They stack several layers to achieve better performance. At last, the $\mathbf{h}_i^{(K)}$ is employed in a sequence input of a Bi-LSTM in depth-first order to the final node representation.

0.5.2 Graph Neural Networks for Multi-relational Graphs

In practice, many graphs have various edge types, such as knowledge graph, AMR graph, etc., which can be formalized as multi-relational graphs. Formally, for a graph $\mathcal{G}$, s.t. $|\mathcal{T}| = 1$ and $|\mathcal{R}| > 1$ is defined as multi-relational graphs. In this section, we introduce different techniques for representing and learning the multi-relational graphs. Specifically, in Section. 0.5.2, we will discuss the formalization for the multi-relational graphs from an original heterogeneous graph. In Section. 0.5.2 and 0.5.2, we will discuss the basic graph representation learning methods and transformers for relational heterogeneous, respectively (we denote it as multi-relational graph neural network for simplification).

Multi-relational Graph Formalization

Since heterogeneous graphs are commonly observed in NLP domian, such as knowledge graph, AMR graph, etc, most of the researchers (Guo *et al.*, 2019c; Ribeiro *et al.*, 2019a; Beck *et al.*, 2018a; Damonte and Cohen, 2019; Koncel-Kedziorski *et al.*, 2019) propose to convert it to a multi-relational

graph, which can be learned by relational GNN in Section. 0.5.2 and Section. 0.5.2.

As defined before, the multi-relational graph is denoted as $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{R})$, s.t. $|\mathcal{T}| = 1$ and $|\mathcal{R}| >= 1$. To get the multi-relational graph, technically, they ignore node types (i.e., project the nodes to the unified embedding space regardless of original nodes or relational nodes). As for edges, they assign the initial edges with the type "default". For each edge $e_{i,j}$, they add a reverse edge $e_{j,i}$ with type "reverse". Besides, for each node v_i, they add the self-loops with the type "self". Thus the converted graph is the multi-relational graph with $|E| = 3$ and $|V| = 1$.

Multi-relational Graph Neural Networks

The multi-relational GNN is the extension of classic GNN for multi-relational graphs, which has the same node type but different edge types. They are originally introduced to encode relation-specific graphs such as knowledge graphs (Schlichtkrull *et al.*, 2018; Malaviya *et al.*, 2020) and parsing graphs (Beck *et al.*, 2018a; Song *et al.*, 2019), which have complicated relationships between nodes with the same type. Generally, most multi-relational GNNs employ type-specific parameters to model the relations individually. In this subsection, we will introduce the classic relational GCN (R-GCN) (Schlichtkrull *et al.*, 2018), relational GGNN (R-GGNN) (Beck *et al.*, 2018a) and relational GAT (R-GAT) (Wang *et al.*, 2020b; Wang *et al.*, 2020h).

R-GCN The R-GCN (Schlichtkrull *et al.*, 2018) is explicitly developed to handle highly multi-relational graphs, especially knowledge bases. The R-GCN is a natural extension of the message-passing GCN framework (Gilmer *et al.*, 2017) which operates on local graph neighborhoods. They group the incoming nodes according to the label types and then apply messaging passing separately. Thus, the aggregation of node v_i's immediate neighbor nodes is defined as

$$\mathbf{h}_i^{(k)} = \sigma(\sum_{r \in \mathcal{E}} \sum_{v_j \in N_r(v_i)} \frac{1}{c_{i,r}} \mathbf{W}_r^{(k)} \mathbf{h}_j^{(k-1)} + \mathbf{W}_0^{(k)} \mathbf{h}_i^{(k-1)}), \tag{34}$$

where $\mathbf{W}_r^{(k)} \in \mathbb{R}^{d \times d}$, $\mathbf{W}_0^{(k)} \in \mathbb{R}^{d \times d}$ are trainable parameters, $N_r(v_i)$ is the neighborhoods of node v_i with relation $r \in \mathcal{E}$, $c_{i,r}$ is the problem-specific

normalization scalar such as $|N_r(v_i)|$, and $\sigma(\cdot)$ is the ReLU activation function. Intuitively, such a step projects neighbor nodes with different relations by relation-specific transformation to unified feature space and then accumulates them through a normalized sum. The self-connection with a special relation type is added to ensure the node itself feature can be held.

However, modeling the multi-relations using separate parameters for each relation can lead to severe over-parameterization, especially on the rare relations. Thus two regularization methods: *basis* and *basis-diagonal*-decomposition are proposed to address this issue. Firstly, for *basis* decomposition, each relation weight $\mathbf{W}_r^{(k)}$ is defined as follows:

$$\mathbf{W}_r^{(k)} = \sum_{b=1}^{B} a_{rb}^{(k)} \mathbf{V}_b^{(k)}, \tag{35}$$

where $\mathbf{V}_b^{(k)} \in \mathbb{R}^{d \times d}$ is the basis and $a_{rb}^{(k)}$ is the associated coefficients. This strategy actually regards the relation matrices as the linear combination of shared basis, which can be seen as a form of weight sharing between different relations.

In the *basis-diagonal decomposition*, each $\mathbf{W}_r^{(k)}$ is defined through the direct sum over a set of low-dimensional matrices as

$$\mathbf{W}_r^{(k)} = \bigoplus_{b=1}^{B} \mathbf{Q}_{br}^{(k)}, \tag{36}$$

where $\mathbf{Q}_{br}^{(k)} \in \mathbb{R}^{d/B \times d/B}$ is the low-dimensional matrix. Thereby, the $\mathbf{W}_r^{(k)}$ is represented by a set of sub matrices as $diag(\mathbf{Q}_{1r}^{(k)}, \mathbf{Q}_{2r}^{(k)}, ..., \mathbf{Q}_{Br}^{(k)})$. This strategy can be seen as a matrix sparsity constraint. It holds the hypothesis that the latent features can be represented by sets of variables that are more tightly coupled within groups than across groups.

There are also some other GCN-based multi-relational graph neural networks for different purposes. For example, Directed-GCN (Marcheggiani and Titov, 2017) is developed to exploit the syntactic graph, which has massive and unbalanced relations. The basic idea of incorporating edge-type information is similar to the R-GCN (Schlichtkrull *et al.*, 2018), but they solve the over-parameterization issue by sharing projection matrix weights for all edges with the same directions but only keeping the relation-specific biases. The other example is weighted-GCN (Shang *et al.*, 2019), which adopt relation-specific

transformation to learn relational information. The weighted-GCN learns the weight score for each relation type end-to-end and inject it into the GCN framework. In this way, the weighted-GCN is capable of controlling how much information each type contributes to the aggregation procedure. As a combination model, the Comp-GCN (Vashishth *et al.*, 2019) generalizes several of the existing multi-relational GCN methods (i.e., R-GCN (Schlichtkrull *et al.*, 2018), Weighted-GCN (Shang *et al.*, 2019) and Directed-GCN (Marcheggiani and Titov, 2017)) and jointly learn the nodes and relations representation.

R-GGNN The relational GGNN (Beck *et al.*, 2018a) is originally developed for the graph-to-sequence problem. It is capable of capturing long-distance relations. Similarly to R-GCN, R-GGNN uses relation-specific weights to capture relation-specific correlations between nodes better. Thus, the propagation process of R-GGNN can be summarized as

$$\begin{aligned}
\mathbf{r}_i^{(k)} &= \sigma\Big(\sum_{v_j \in N(v_i)} \frac{1}{c_{v_i,r}} \mathbf{W}^r_{\phi(e_{i,j})} \mathbf{h}_j^{(k-1)} + \mathbf{b}^r_{\phi(e_{i,j})}\Big), \\
\mathbf{z}_i^{(k)} &= \sigma\Big(\sum_{v_j \in N(v_i)} \frac{1}{c_{v_i,z}} \mathbf{W}^z_{\phi(e_{i,j})} \mathbf{h}_j^{(k-1)} + \mathbf{b}^z_{\phi(e_{i,j})}\Big), \\
\tilde{h}_i^{(k)} &= \rho\Big(\sum_{v_j \in N(v_i)} \frac{1}{c_{v_i}} \mathbf{W}_{\phi(e_{i,j})} (\mathbf{r}_j^{(k)} \odot \mathbf{h}_i^{(k-1)}) + \mathbf{b}_{\phi(e_{i,j})}\Big), \\
\mathbf{h}_i^{(k)} &= (1 - \mathbf{z}_i^{(k)}) \odot \mathbf{h}_i^{(k-1)} + \mathbf{z}_i^{(k)} \odot \tilde{h}_i^{(k)},
\end{aligned} \tag{37}$$

where $\mathbf{W}^{r/z/\cdot}_{\phi(e_{i,j})} \in \mathbb{R}^{d \times d}, \mathbf{b}^{r/z/\cdot}_{\phi(e_{i,j})}$ are trainable relation-specific parameters, $\sigma(\cdot)$ is the sigmoid function, $c_{v_i,r/z/\cdot} = |N(v_i)|$, and $\rho(\cdot)$ is the non-linear activation function such as tanh and ReLU.

R-GAT Wang *et al.* (2020b) and Wang *et al.* (2020h) propose to extend the classic GAT to fit the multi-relational graphs. In this section, we will discuss two R-GAT variants. Intuitively, neighbor nodes with different relations should have different influences.

Wang *et al.* (2020b) propose to extend the homogeneous GAT with additional relational heads. Technically, they propose the relational node representation as

$$\mathbf{h}_{i,rel}^{(k),m} = \sum_{v_j \in N(v_i)} \beta_{ij}^{(k),m} \mathbf{W}^{(k),m} \mathbf{h}_j^{(k-1)} \tag{38}$$

where $m \in [1, M]$ is the $m-$th head and $\beta_{ij}^{(k),m}$ is the corresponding attention score for relation head m, which is calculated as

$$\begin{aligned} s_{ij}^{(k),m} &= f(\mathbf{e}_{i,j}), \\ \beta_{ij}^{(k),m} &= \text{softmax}_j(s_{ij}^{(k),m}), \end{aligned} \tag{39}$$

where $s_{ij}^{(k),m}$ is the similarity between node v_i and v_j, and $f(\cdot) : \mathbb{R}^{d^k} \rightarrow \mathbb{R}$ is the multi-layer transformation (MLP) with non-linearity. The relational representation of node v_i is the concatenation with linear transformation of M heads' results as:

$$\mathbf{h}_{i,rel}^{(k)} = g(||_{m=1}^{M} \mathbf{h}_{i,rel}^{(k),m}), \tag{40}$$

where $||$ denotes the vector concatenation operation and $g(\cdot) : \mathbb{R}^{m \times d^k} \rightarrow \mathbb{R}^{d^k}$ is the liner projection. Thus, the final node representation is the combination of $\mathbf{h}_{i,rel}^{(k)}$ and $\mathbf{h}_{i,att}^{(k)}$ as follows:

$$\mathbf{h}_i^{(k)} = \sigma(\mathbf{W}^{(k)}(\mathbf{h}_{i,rel}^{(k)} || \mathbf{h}_{i,att}^{(k)}) + \mathbf{b}^{(k)}), \tag{41}$$

where $\mathbf{W}^{(k)} \in \mathbb{R}^{d \times d}, \mathbf{b}^{(k)} \in \mathbb{R}^d$ are trainable parameters, $\sigma(\cdot)$ is the ReLU activation function.

Unlike the work by Wang *et al.* (2020b), which learn and fuse the relation-specific node embedding regarding each type of edges, Wang *et al.* (2020h) develops a relation-aware attention mechanism to calculate the attention score $\alpha_{ij}^{(k),m}$ as

$$\begin{aligned} \alpha_{ij}^{(k),m} &= \text{softmax}_j(s_{ij}^{(k),m}), \\ s_{ij}^{(k),m} &= \sigma(f^{(k),m}([\mathbf{W}^{(k),m}\mathbf{h}_i^{(k-1)}; \mathbf{W}^{(k),m}\mathbf{h}_j^{(k-1)}; \mathbf{e}_{i,j}^{(k-1)}])), \end{aligned} \tag{42}$$

where $\mathbf{W}^{(k),m} \in \mathbb{R}^{d \times d}$ is the learnable matrix, and $f(\cdot)^{(k),m} : \mathbb{R}^{3 \times d} \rightarrow \mathbb{R}$ is the single linear projection layer. They learn a global representation for each relation type $r = \phi(e_{i,j}) \in \mathcal{R}$. Technically, for all edges with type $r \in \mathcal{R}$, two node sets S_r and T_r. S_r are collected regarding the source and target node set of relation r, respectively. Thus the edge type embedding $\mathbf{t}_r$ can be calculated by:

$$\mathbf{e}_r = |\frac{\sum_{o \in S_r} \mathbf{W}\mathbf{h}_o}{|S_r|} - \frac{\sum_{o \in T_r} \mathbf{W}\mathbf{h}_o}{|T_r|}|. \tag{43}$$

Thus the edge representation is the absolute difference between mean vectors of source and target nodes connected by edges whose type are r.

Gating Mechanism The multi-relational graph neural networks also face the over-smoothing problem when stacking several layers to exploit implicit correlations between distant neighbors (i.e., not directly connected) (Tu *et al.*, 2019a). To solve this, the gating mechanism, which combines the nodes' input features and aggregated features by gates, is introduced to the multi-relational graph neural network (Tang *et al.*, 2020c; De Cao *et al.*, 2018; Tu *et al.*, 2019a; Cao *et al.*, 2019c). Intuitively, the gating mechanism can be regarded as a trade-off between the original signals and the learned information. It regulates how much of the update message that are propagated to the next step, thus preventing the model from thoroughly overwriting the past information. Here we introduce the gating mechanism by taking the classic R-GCN (Schlichtkrull *et al.*, 2018) as an example, which actually can be extended to arbitrary variants.

We denote the representation before activation $\sigma(\cdot)$ as

$$\mathbf{u}_i^{(k)} = f^{(k)}(\mathbf{h}_i^{(k-1)}), \tag{44}$$

where f denotes to the aggregation function. Ultimately, the final representation of node i's representation is a gated combination of the previous embedding $\mathbf{h}_i^k$ and GNN output representation $\sigma(\mathbf{u}_i^k)$ as:

$$\mathbf{h}_i^{(k)} = \sigma(\mathbf{u}_i^{(k)}) \odot \mathbf{g}_i^{(k)} + \mathbf{h}_i^{(k-1)} \odot (1 - \mathbf{g}_i^{(k-1)}) \tag{45}$$

where $\mathbf{g}_i^k$ is the gating vectors, and $\sigma(\cdot)$ is often the *tanh*$(\cdot)$ function. The gating vectors are calculated by both the inputs and outputs as follows:

$$g_i^{(k)} = \sigma(f^{(k)}([\mathbf{u}_i^{(k)}, \mathbf{h}_i^{(k-1)}])) \tag{46}$$

where σ is the sigmoid activation function, and $f^{(k)}(\cdot) : \mathbb{R}^{2d} \to \mathbb{R}$ is the linear transformation. We repeat calculating $g_i^{(k)}$ for d times to get the gating vector $\mathbf{g}_i^{(k)}$.

Graph Transformer

The transformer architecture (Vaswani *et al.*, 2017) has achieved great success in NLP fields. Roughly speaking, the transformer's self-attention mechanism is a special procedure of fully connected implicit graph learning, as we discussed in sec. 0.4.2, thus bridging the concept of GNN and transformer. However,

the traditional transformer fails to leverage structure information. Inspired by GAT (Velickovic *et al.*, 2018), which combines the message passing with attention mechanism, much literature incorporates structured information to the transformer (we name it as graph transformer) by developing structure-aware self-attention mechanism (Yao *et al.*, 2020; Levi, 1942; Xiao *et al.*, 2019; Zhu *et al.*, 2019b; Cai and Lam, 2020b; Wang *et al.*, 2020f). In this section, we will discuss the techniques of structure-aware self-attention for the multi-relational graph.

As a preliminary knowledge, here we give a brief review of self-attention. To make it clear, we omit the multi-head mechanism and only present the self-attention function. Formally, we denote the input of self-attention as $\mathbf{Q} = \{\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_m\} \in \mathbb{R}^{m \times d^q}, \mathbf{K} = \{\mathbf{k}_1, \mathbf{k}_2, ..., \mathbf{k}_n\} \in \mathbb{R}^{n \times d^k}, \mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n\} \in \mathbb{R}^{n \times d^v}$. Then the output representation $\mathbf{z}_i$ is calculated as

$$\mathbf{z}_i = \text{Attention}(\mathbf{q}_i, \mathbf{K}, \mathbf{V}) = \sum_{j=1}^{n} \alpha_{i,j} \mathbf{W}^v \mathbf{v}_j \tag{47}$$

$$\alpha_{i,j} = \text{softmax}_j(u_{i,j}) \tag{48}$$

$$u_{i,j} = \frac{(\mathbf{W}^q \mathbf{q}_i)^T (\mathbf{W}^k \mathbf{k}_j)}{\sqrt{d}} \tag{49}$$

where $\mathbf{W}^q \in \mathbb{R}^{d \times d^q}, \mathbf{W}^k \in \mathbb{R}^{d \times d^k}, \mathbf{W}^v \in \mathbb{R}^{d \times d^v}$ are trainable parameters, and d is the model dimension. Note that for graph transformer, the query, key and value all refer to the nodes' embedding:, namely, $\mathbf{q}_i = \mathbf{k}_i = \mathbf{v}_i = \mathbf{h}_i$. Thus, we will only use $\mathbf{h}_i$ to represent query, key and value considering simplification in the following contents.

There are various graph transformers for relation graphs that incorporate the structure knowledge, which can be categorized into two classes according to the self-attention function. One class is the R-GAT-based methods which adopt relational GAT-like feature aggregation. Another class reserves the fully connected graph while incorporating the structure-aware relation information to the self-attention function.

R-GAT Based Graph Transformer. The GAT-based graph transformer (Yao *et al.*, 2020) adopts the GAT-like feature aggregation, which leverages the graph connectivity inductive bias. Technically, they first aggregate neighbors

with type-specific aggregation step and then fuse them through feed-forward layer as follows:

$$\begin{aligned}\mathbf{z}_i^{r,(k)} &= \sum_{v_j \in N_r(v_i)} \alpha_{i,j}^k \mathbf{W}^{v,(k)} \mathbf{h}_j^{(k-1)}, r \in \mathcal{E} \\ \mathbf{h}_i^{(k)} &= \text{FFN}^{(k)}(\mathbf{W}^{O,(k)}[\mathbf{z}_i^{R_1,(k)}, ..., \mathbf{z}_i^{R_q,(k)}]),\end{aligned} \tag{50}$$

where $\text{FFN}^{(k)}(\cdot)$ denotes the feed-forward layer in transformer (Vaswani *et al.*, 2017), and $\alpha_{i,j}$ denotes the dot-product score in eq. 49.

To incorporate the bi-directional information, Wang *et al.* (2020f) learns forward and backward aggregation representation in graph transformer. Specifically, given the backward and forward features (i.e., $\mathbf{h}_{i,\vdash}$ and $\mathbf{h}_{i,\dashv}$) for node v_i, the backward aggregated feature $\mathbf{z}_{i,\dashv}^{(k)}$ for node v_i is formulated by:

$$\begin{aligned}\mathbf{z}_{i,\dashv}^{(k)} &= \sum_{v_j \in N_\dashv(v_i)} \alpha_{i,j} \mathbf{W}^{v,(k)} \mathbf{a}_{i,j}^{(k)}, \\ \mathbf{a}_{i,j}^{(k)} &= f^{(k)}([\mathbf{h}_{i,\vdash}, \mathbf{e}_{i,j}; \mathbf{h}_{j,\dashv}]),\end{aligned} \tag{51}$$

where $f^{(k)}(\cdot) : \mathbb{R}^{3\times d \to \mathbb{R}^d}$ is the linear transformation, and $\alpha_{i,j}$ is the softmax score of incoming neighbors' dot-production results $u_{i,j}$ which can be formulated by:

$$u_{i,j} = \frac{(\mathbf{W}^{q,(k)} \mathbf{h}_{i,\dashv})^T (\mathbf{W}^{k,(k)} \mathbf{a}_{i,j}^{(k)})}{\sqrt{d}}. \tag{52}$$

Then they adopt the gating mechanism to fuse bidirectional aggregated features to get the packaged node representation:

$$\begin{aligned}\mathbf{g}_i^{(k)} &= \sigma(f'^k([\mathbf{z}_{i,\vdash}^{(k)}; \mathbf{z}_{i,\dashv}^{(k)}])), \\ \mathbf{p}_i^{(k)} &= \mathbf{g}_i^{(k)} \odot \mathbf{z}_{i,\vdash}^{(k)} + (1 - \mathbf{g}_i^{(k)}) \odot \mathbf{z}_{i,\dashv}^{(k)}\end{aligned} \tag{53}$$

where $f'(\cdot) : \mathbb{R}^{2\times d} \to \mathbb{R}^d$, and $\sigma(\cdot)$ is the sigmoid activation function. They calculate the the forward and backward node representation based on the packaged representation, respectively:

$$\begin{aligned}[\mathbf{o}_{i,\vdash}^{(k)}, \mathbf{o}_{i,\dashv}^{(k)}] &= \text{FFN}^{(k)}(\mathbf{p}_i^{(k)}), \\ \mathbf{h}_{i,\vdash}^{(k)} &= \text{LayerNorm}^{(k)}(\mathbf{o}_{i,\vdash}^{(k)} + \mathbf{h}_{i,\vdash}^{(k-1)}), \\ \mathbf{h}_{i,\dashv}^{(k)} &= \text{LayerNorm}^{(k)}(\mathbf{o}_{i,\dashv}^{(k)} + \mathbf{h}_{i,\dashv}^{(k-1)}),\end{aligned} \tag{54}$$

where $\text{FFN}(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^{2\times d}$ is the feed-forward function, and $\text{LayerNorm}(\cdot)$ is the layer normalization (Ba *et al.*, 2016). The final node representation is the concatenation of the last layer K's bidirectional representations:

$$\mathbf{h}_i^{(K)} = f''^K([\mathbf{h}_{i,\vdash}^{(K)}, \mathbf{h}_{i,\dashv}^{(K)}]), \tag{55}$$

where $f''^{(K)}(\cdot) : \mathbb{R}^{2\times d} \rightarrow \mathbb{R}^d$ is the linear transformation.

Structure-aware Self-attention Based Graph Transformer. Unlike the R-GAT-based graph transformer, which purely relies on the given graph structure as connectivity, the structure-aware self-attention-based graph transformer reserves the original self-attention architecture, allowing non-neighbor nodes' communication. We will firstly discuss the structure-aware self-attention mechanism and then present its unique edge embedding representation.

Shaw *et al.* (2018) firstly attempts to model the relative relations between words (nodes) in the neural machine translation task. Technically, they consider the relation embedding when calculating node-wise similarity in eq. 49 as follows:

$$u_{i,j}^{(k)} = \frac{(\mathbf{W}^{q,(k)}\mathbf{h}_i^{(k-1)})^T(\mathbf{W}^{k,(k)}\mathbf{h}_j^{(k-1)}) + (\mathbf{W}^{q,(k)}\mathbf{h}_i^{(k-1)})^T\mathbf{e}_{i,j}}{\sqrt{d}}. \tag{56}$$

Motivated by Shaw *et al.* (2018), Xiao *et al.* (2019) and Zhu *et al.* (2019b) propose to extend the conventional self-attention architecture to explicitly encode the relational embedding between nodes pairs in the latent space as

$$\begin{aligned} u_{i,j}^{(k)} &= \frac{(\mathbf{W}^{q,(k)}\mathbf{h}_i^{(k-1)})^T(\mathbf{W}^{k,(k)}\mathbf{h}_j^{(k-1)} + \mathbf{W}^{r,(k)}\mathbf{e}_{i,j})}{\sqrt{d}}, \\ \mathbf{h}_i^{(k)} &= \sum_{j=1}^{n} \alpha_{i,j}^k(\mathbf{W}^{v,(k)}\mathbf{h}_j^{(k-1)} + \mathbf{W}^{f,(k)}\mathbf{e}_{i,j}). \end{aligned} \tag{57}$$

To adopt the bidirectional relations, Cai and Lam (2020b) extends the traditional self-attention as follows:

$$u_{i,j}^{(k)} = \frac{[\mathbf{W}^{q,(k)}(\mathbf{h}_i^{(k-1)} + \mathbf{e}_{i,j})]^T[\mathbf{W}^{k,(k)}(\mathbf{h}_j^{(k-1)} + \mathbf{e}_{j,i})]}{\sqrt{d}}. \tag{58}$$

Given the learnt attention for each relation, edge embedding representation is the next critical step for incorporating the structure-information. Shaw *et al.*

(2018) simply learns the relative position encoding w.r.t the nodes' absolute positions. Technically, they employ $2K+1$ latent labels ($[-K, K]$) and project $j - i$ to one specific label embedding for node pair (v_i, v_j) to fetch the edge embedding $\mathbf{e}_{i,j}$. Xiao *et al.* (2019) adopts the similar idea as Shaw *et al.* (2018). They define a relative position embedding table and fetch the edge embedding by looking up from it.

Zhu *et al.* (2019b) and Cai and Lam (2020b) learn the edge representation $\mathbf{e}_{i,j}$ by the path from node v_i to node v_j. For Zhu *et al.* (2019b), the natural way is to view the path as a string, which is added to the vocabulary to vectorize it. Other ways are further proposed to learn from labels' embedding along the path, such as 1) taking average, 2) taking sum, 3) encoding them using self-attention, and 4) encoding them using CNN filters. Cai and Lam (2020b) propose the shortest path based relation encoder. Concretely, they firstly fetch the labels' embedding sequence along the path. Then they employ the bi-directional GRUs for sequence encoding. The last hidden states of the forward and backward GRU networks are finally concatenated to represent the relation embedding $\mathbf{e}_{i,j}$.

0.5.3 Graph Neural Networks for Heterogeneous Graph

In practice, many graphs have various node and edge types, such as knowledge graph, AMR graph, etc., which are called heterogeneous graphs. Formally, for a graph $\mathcal{G}$, s.t. $|\mathcal{T}| > 1$ or $|\mathcal{R}| > 1$, it is called heterogeneous graph. Beside transforming the heterogeneous to relation graphs, as introduced in the previous subsection, sometimes it is required to fully leverage the type information for both nodes and edges (Hu *et al.*, 2020c; Fan *et al.*, 2019; Feng *et al.*, 2020a; Wang *et al.*, 2020a; Linmei *et al.*, 2019; Zhang *et al.*, 2019b). Thus, in Section. 0.5.3, we first introduce a pre-processing technique for heterogeneous graph. Then, in Section. 0.5.3 and 0.5.3, we will introduce two typical graph representation learning methods specially for heterogeneous graphs.

Levi Graph Transformation

Since most existing GNN methods are only designed for homogeneous conditions and there is a massive computation burden when dealing with lots of edge types (Beck *et al.*, 2018a) (e.g. an AMR graph may contain more than

100 edge types), it is typical to effectively to treat the edges as nodes in the heterogeneous graphs (Beck *et al.*, 2018a; Xu *et al.*, 2018c; Sun *et al.*, 2019a; Guo *et al.*, 2019c).

One of the important graph transformation techniques is *Levi Graph Transformation*. Technically, for each edge $e_{i,j}$ with edge label $\phi(e_{i,j})$, we will create a new node $v_{e_{i,j}}$. Thus the new graph is denoted as $\mathcal{G}'(\mathcal{V}', \mathcal{E}', \mathcal{T}', \mathcal{R}')$, where the node set is $\mathcal{V}' = \mathcal{V} \cup \{v_{e_{i,j}}\}$, the node label set is $\mathcal{T}' = \mathcal{T} \cup \{\phi(e)_{i,j}\}$. We cut off the direct edge between node v_i, v_j and the add two direct edges between: 1) $v_i, v_{e_{i,j}}$, and 2) $v_{e_{i,j}}, v_j$. After converting all edges in $\mathcal{E}$, the new graph $\mathcal{G}'$ will be a bipartite graph, s.t. $|\mathcal{R}'| = 1$. An example of transforming AMR graph to desired levi-graph is illustrated in Fig. 11. The obtained graph is a simplified heterogeneous levi graph that has a single edge type but unrestricted node types, which can then be learnt by heterogeneous GNNs described in Section 0.5.3.

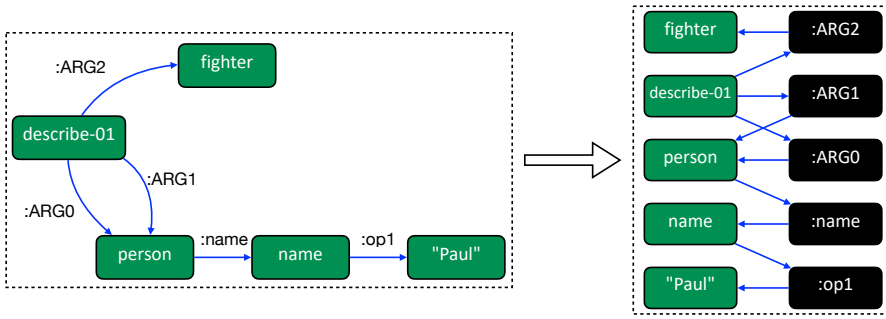


Figure 11: An example for transforming AMR graph to Levi-graph.

Meta-path based Heterogeneous GNN

Meta-path, a composite relation connecting two objects, is a widely used structure to capture the semantics. Take movie data IMDB for example, there are three types of nodes, including movie, actor, and director. The meta-path $Movie \rightarrow Actor \rightarrow Movie$, which covers two movie sets and one actor, describes the co-actor relations. Thus different relations between nodes in the heterogeneous graph can be easily revealed by meta-paths.

First, we provide the meta-level (i.e., schema-level) description of a heterogeneous graph for better understanding. We follow the setting of heterogeneous

information network (HIN) (Sun *et al.*, 2011) and give the concept of Network Schema. The network schema is a meta template for the heterogeneous graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with the node type mapping: $\mathcal{V} \rightarrow \mathcal{T}$ and edge type mapping: $\mathcal{E} \rightarrow \mathcal{R}$. We denote it as $\mathcal{M}_{\mathcal{G}}(\mathcal{T}, \mathcal{R})$. A meta path is a path on the network schema denoted as $\Phi = T_1 \overset{R_1}{\rightarrow} T_2 \overset{R_2}{\rightarrow} ... \overset{R_l}{\rightarrow} T_{l+1}$, where $T_i \in \mathcal{T}$ is the schema's node and $R_i \in \mathcal{R}$ is the corresponding relation node. What's more, we denote the meta-path set as $\{\Phi_1, \Phi_2, ..., \Phi_p\}$. For each node T_i on the meta-path Φ_j, we denote it as $T_i^{\Phi_j}$. Then we combine the network schema with the concrete heterogeneous graph. For each node v_i in the heterogeneous graph and one meta-path Φ_j, we define the meta-path-based neighbors as $N_{\Phi_j}(v_i)$, which contains all nodes including itself linked by meta-path Φ_j. An example of meta-path based heterogeneous graph is shown in Fig. 12. Conceptually, the neighbor set can have multi-hop nodes depending on the length of the meta-path.

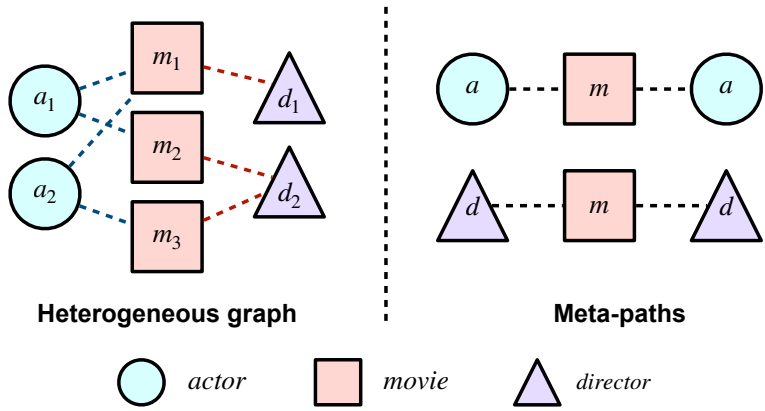


Figure 12: An example of meta-path based heterogeneous graph.

Most meta-path-based GNN methods adopt the attention-based aggregation strategy (Wang *et al.*, 2020a; Fan *et al.*, 2019). Technically, they can be generally divided into two stages. Firstly, they aggregate the neighborhoods along each meta-paths, which can be named as "node-level aggregation". After this, the nodes receive neighbors' information via different meta-path. Next, they apply meta-path level attention to learn the semantic impact of different meta-path. In this way, they can learn the optimal combination of neighbors connected by multiple meta-paths. In the following, we will discuss two typical

heterogeneous GNN models (Wang *et al.*, 2020a; Fan *et al.*, 2019).

HAN (Wang *et al.*, 2019e) Due to the heterogeneity of nodes in graphs, different nodes have different feature spaces, which brings a big challenge for GNN to handle various initial node embedding. To tackle this issue, the type-specific transformation is adopted to project various nodes to a unified feature space as follows:

$$\mathbf{h}_i' = \mathbf{W}_{\tau(v_i)}\mathbf{h}_i. \tag{59}$$

We overwrite the notation $\mathbf{h}_i$ to denote the transformed node embedding in HAN's discussion.

- **Node-level Aggregation.** For the node v_i and its' neighbor node set $N_{\Phi_k}(v_i)$ on the meta-path Φ_k, the aggregated feature of node v_i can be represented by:

$$\begin{aligned}
\mathbf{z}_{i,\Phi_k} &= \sigma\Big(\sum_{v_j \in N_{\Phi_k}(v_i)} \alpha_{i,j}^{\Phi_k}\mathbf{h}_j\Big) \\
\alpha_{i,j}^{\Phi_k} &= \text{softmax}_j(u_{i,j}^{\Phi_k}) \\
u_{i,j}^{\Phi_k} &= \text{Attention}(\mathbf{h}_i, \mathbf{h}_j; \Phi_k) = \sigma(\mathbf{W}[\mathbf{h}_i, \mathbf{h}_j]),
\end{aligned} \tag{60}$$

  where $\mathbf{W} \in \mathbb{R}^{1\times 2d}$ is the trainable parameter. To make the training process more stable, they further extend the attention by the multi-head mechanism. The final representation of $\mathbf{z}_{i,\Phi_j}$ is the concatenation of L heads' results. Given p meta-paths, we can obtain the aggregated embedding via the previous nodel-level aggregation step as $\{\mathbf{Z}_{\Phi_1}, ..., \mathbf{Z}_{\Phi_p}\}$, where $\mathbf{Z}_{\Phi_j}$ is the collection of all nodes' representation for meta-path Φ_j.

- **Meta-path Level Aggregation.** Generally, different meta-path conveys different semantic information. To this end, they aim to learn the importance of each meta-path as:

$$(\beta_{\Phi_1}, ..., \beta_{\Phi_p}) = \text{Meta\_Attn}(\mathbf{Z}_{\Phi_1}, ..., \mathbf{Z}_{\Phi_p}), \tag{61}$$

  where β_{Φ_j} denotes the learned importance score for meta-path Φ_j, and Meta_Attn() is the attention-based scorer. Technically, for each meta-path, they first learn the semantic-level importance for each node. Then

they average them to get the meta-path level importance. It can be formulated by:

$$o_{\Phi_k} = \frac{1}{|\mathcal{V}|} \sum_{v_i \in \mathcal{V}} \mathbf{q}^T f(\mathbf{z}_{i,\Phi_k}),$$
$$\beta_{\Phi_k} = \text{softmax}_k(o_{\Phi_k}), \tag{62}$$

where $f(\cdot) : \mathbb{R}^d \to \mathbb{R}^d$ is the MLP with tanh non-linearity.

Finally, we can obtain the final node representation as:

$$\mathbf{z}_i = \sum_{k=1}^{p} \beta_{\Phi_k} \mathbf{z}_{i,\Phi_k}. \tag{63}$$

MEIRec The MEIRec (Fan *et al.*, 2019) is a heterogeneous GNN-based recommend system. In the specific recommendation system condition, they restrict the heterogeneous graph's type amount and meta-path's length and propose a special heterogeneous graph neural network, particularly to fully utilize rich structure information. Considering that the type-specific transformation requires huge parameters when the amount of nodes is large, they propose an efficient unified embedding learning method. For each node, they fetch the terms in the vocabulary and then average them to get the vector representation.

- **Node-level Aggregation.** Unlike HAN (Hu *et al.*, 2020c), which collect all nodes along the meta-path as neighbors, they treat different hop of neighbors differently. Given the meta-path Φ_j, they define the neighbors of node v_i as $N_{\Phi_j}(v_i)^o, o \in [1, 2, ..., O]$ where o denotes the hop. They learn the representation recursively. Take $(o)-$hop neighbors for example, for each nodes in $N_{\Phi_j}(v_i)^o$, they first collect the immediate-neighbors from $(o+1)$-hop neighbors and learn the representation to obtain (o)-hop representation. Then they repeat this procedure to obtain $(o-1)$-hop nodes' representation. Finally, v_i's representation for meta-path Φ_j is generated. Formally, for $v_k \in N_{\Phi_j}(v_i)^o$, they define its' immediate neighbor set as $N_{\Phi_j}(v_k) \in N_{\Phi_j}(v_i)^{o+1}$. Node v_j's representation $\mathbf{z}_{k,\Phi_j}$ is formulated as:

$$\mathbf{z}_{k,\Phi_j} = g(\{\mathbf{z}_{l,\Phi_j}, v_l \in N_{\Phi_j}(v_k)\}), \tag{64}$$

where $g(\cdot)$ is the aggregation function. In MEIRec, it can be the average function, LSTM function, or the CNN function depend on the nodes'

type. Besides, the last hop ((O)-hop)'s nodes are represented by initial node embedding.

- **Meta-path Level Aggregation.** Given p meta-path with the starting nodes v_i, we can obtain p aggregated embedding by the previous step. Then we adopt a similar procedure as node-level aggregation as follows:

$$\mathbf{z}_i = g(\{\mathbf{z}_{i,\Phi_j}, j \in [1, ..., p]\}), \tag{65}$$

where $g(\cdot)$ is the aggregation function, as we discussed before.

R-GNN based Heterogeneous GNN

Although the meta-path is an effective tool to organize the heterogeneous graph, it requires additional domain expert knowledge. To this end, most researchers adopt a similar idea from R-GNN by using type-specific aggregation. For clarity, we name these methods as R-GNN based heterogeneous GNN and introduce several typical variants of this category in the following.

HGAT HGAT (Linmei *et al.*, 2019) is proposed for encoding heterogeneous graph which contains various node types but single edge types. In other words, the edge only represents connectivity. Intuitively, for a specific node, different types of neighbors may have different relevance. To fully exploit diverse structure information, HGAT firstly focuses on global types' relevance learning (type-level learning) and then learns the representation for specific nodes (node-level learning).

- **Type-level learning**. Technically, for a specific node v_i and its' neighbors $N(v_i)$, HGAT get the type-specific neighbor representation as:

$$\mathbf{z}_t^{(k)} = \sum_{v_j \in N_t(v_i)} \mathbf{h}_j^{(k-1)}, t \in \mathcal{T}. \tag{66}$$

Note that we overwrite $N_t(v_i)$ which denotes the neighbors with node type t. Then they calculate the relevance of each type by attention mechanism:

$$\begin{aligned} s_t &= \sigma(\mathbf{q}^T[\mathbf{h}_i^{(k-1)}, \mathbf{z}_t^{(k)}]), \\ \alpha_t &= \frac{exp(s_t)}{\sum_{t' \in \mathcal{T}} exp(s_{t'})}, \end{aligned} \tag{67}$$

where $\mathbf{q}$ is the trainable vector.

- **Node-level learning** Secondly, they apply R-GCN (Schlichtkrull *et al.*, 2018) like aggregation procedure for a different type of nodes. Formally, for the node v_i and the type relevance scores $\{\alpha_t\}, t \in \mathcal{T}$, HGAT calculate each neighbors' attention score as follows:

$$\begin{aligned} b_{i,j} &= \sigma(\mathbf{q}_1^T \alpha_{\tau(v_j)}[\mathbf{h}_i^{(k-1)}, \mathbf{h}_j^{(k-1)}]), \\ \beta_{i,j} &= \frac{exp(b_{i,j})}{\sum_{v_m \in N(v_i)} exp(b_{i,m})}, \end{aligned} \tag{68}$$

where $\mathbf{q}_1$ is the trainable vector. Finally, HGAT applies layer-wise heterogeneous GCN to learn the node representation, which is formulated as:

$$\mathbf{h}_i^{(k)} = \sigma(\sum_{t \in \mathcal{T}} \sum_{v_j \in N_t(v_i)} \mathbf{W}_t^{(k)} \mathbf{h}_j^{(k-1)}). \tag{69}$$

MHGRN MHGRN (Feng *et al.*, 2020a) is an extension of R-GCN, which can leverage multi-hop information directly on heterogeneous graphs. Generally, it borrows the idea of relation path (e.g., meta-path) to model the relation of two not k-hop connected nodes and extend the existing R-GNN to the path-based heterogeneous graph representation learning paradigm. The K-hop relation path between node v_i, v_j is denoted as:

$$\Phi_k = \{(v_i, e_{i,1}, ..., e_{k-1,j}, v_j) | (v_i, e_{i,1}, v_1), ..., (v_{k-1}, e_{k-1,j}) \in \mathcal{E}\}. \tag{70}$$

Note that the heterogeneous graph may contain more than one k-hop relation path.

- **k-hop feature aggregation**. First, to make the GNN aware of the node type, they project the nodes' initial feature to the unified embedding space by type-specific linear transformation (the same as eq. 59). Considering simplification, we overwrite nodes' feature notation $\mathbf{h}$ to represent the unified features. Then given node v_i, they aim to aggregate k-hop ($k \in [1, K]$) neighbors' feature as follows:

$$\mathbf{z}_i^{\Phi_k} = \sum_{(v_j, e_{j,1}, ..., e_{k-1,i}, v_i) \in \Phi_k} \frac{\alpha(v_j, e_{j,1}, ..., e_{k-1,i}, v_i)}{\sum_{(v_j, ..., v_i) \in \Phi_k} \alpha(v_j, ..., v_i)} \prod_{l=1}^{l=K} \prod_{o=1}^{o=K} \mathbf{W}_{r_o}^l \mathbf{h}_j, \quad (1 \le k \le K) \tag{71}$$

where $\mathbf{W}^l_{r_o}, 1 \leq l \leq K, 1 \leq o \leq K$ is the learnable matrix, r_o denotes $o-$th hop's edge, $\alpha(j, v_1, ..., v_k, v_i)$ is the attention score among all k-hop paths between node v_j and v_i. We use $\mathbf{z}_i$ to denote the learned embedding for node v_i.

- **Fusing different relation paths**. Next, they fuse relation paths with different length via attention mechanism:

$$\mathbf{z}'_i = \sum_{k=1}^{K} \text{Attention}(\mathbf{q}, \mathbf{z}_i^{\Phi_k})\mathbf{z}_i^{\Phi_k}, \tag{72}$$

where $\mathbf{q}$ is the task-specific vector (in MHGRN, it is the text-based query vector), $\text{Attention}() : \mathbb{R}^d \rightarrow \mathbb{R}$ is the normalized attention score. Note that we omit the details of $\alpha(j, v_1, ..., v_k, v_i)$ and $\text{Attention}()$ since they are task-specific functions. At last, the final representation of node v_i is the shortcut connection between $\mathbf{z}_i$ and original feature $\mathbf{h}_i$ as follows:

$$\mathbf{z}_i = \sigma(\mathbf{W}_1\mathbf{z}'_i + \mathbf{W}_2\mathbf{h}_i), \tag{73}$$

where $\mathbf{W}_1, \mathbf{W}_2$ is the learnable weights.

HGT The HGT (Hu *et al.*, 2020c) is the graph transformer for heterogeneous graphs, which build meta-relations among nodes on top of the network schema, as we discuss in the meta-path-based heterogeneous GNN paragraph. Unlike most previous works that assume the graph structure is static (i.e., time-independent), they propose a relative temporal encoding strategy to learn the time dependency.

The meta-relation is a triple based on the network schema. For each edge $e_{i,j}$ which links node v_i and v_j, the meta-relation is defined as $\Phi_{v_i,e_{i,j}v_j} =< \tau(v_i), \phi(e_{i,j}), \tau(v_j) >$. To further represent the time-dependent relations, they add the timestamps to the start nodes when adding directed edges. Generally, the GNN is defined as:

$$\mathbf{h}_i^{(k)} = \text{Aggregation}^{(k)}_{v_j \in N(v_i)}(\text{Attention}^{(k)}(v_i, e_{i,j}, v_j)\text{Message}^{(k)}(v_i, e_{i,j}, v_j)), \tag{74}$$

where $N(v_i)$ denotes the incoming nodes. We will briefly discuss three basic meta-relation based operations: attention, message message passing, and aggregation, as well as the relative temporal encoding strategy.

- **Attention operation**. The Attention$(\cdot,\cdot,\cdot)$ operation is the mutual attention that calculates the weight of two connected nodes grounded by their meta-relations. Specifically, they employ multi-head attention based on meta-relations, which is formulated as:

$$\begin{aligned}
&\text{Attn_head}^i(v_i, e_{i,j}, v_j) = \\
&\text{f_linear}^i_{\tau(v_i)}(\mathbf{h}_i)\mathbf{W}^{ATT}_{\phi(e_{i,j})}\text{g_linear}^i_{\tau(v_j)}(\mathbf{h}_j)^T \frac{E_{\Phi_{v_i,e_{i,j}v_j}}}{\sqrt{d}}, \\
&\text{Attention}(v_i, e_{i,j}, v_j) = softmax_{v_j \in N(v_i)}(||_{h=1}^{H}\text{Attn_head}^i(v_i, e_{i,j}, v_j))
\end{aligned} \tag{75}$$

  where H is the number of heads, $\text{f\_linear}^i_{\tau(\cdot)}, \text{g\_linear}^i_{\tau(\cdot)} : \mathbb{R}^{d/H} \rightarrow \mathbb{R}^{d/H}$ are the node-type-specific transformation functions for source nodes and target nodes respectively, $\mathbf{W}^{ATT}_{\phi(\cdot)}$ is the edge-type-specific matrix, and $E_{\Phi_{v_i,e_{i,j}v_j}}$ is the meta-path-specific scalar weight.

- **Message passing operation**. The Message$(\cdot)$ is the heterogeneous message passing function. Similar to the Attention$(\cdot,\cdot,\cdot)$ above, they incorporate the meta-relations into the message passing process as follows:

$$\begin{aligned}
\text{msg_head}^i(v_i, e_{i,j}), v_j) &= \text{m_linear}^i_{\tau(v_i)}(\mathbf{h}_i)\mathbf{W}^{MSG}_{\phi(e_{i,j})}, \\
\text{Message}(v_i, e_{i,j}, v_j) &= ||_{h=1}^{H}\text{msg_head}^h(v_i, e_{i,j}, v_j)
\end{aligned} \tag{76}$$

  where $\text{m\_linear}(\cdot) : \mathbb{R}^{d/H} \rightarrow \mathbb{R}^{d/H}$ is the node-type-specific transformation, and $\mathbf{W}^{MSG}_{\phi(\cdot)}$ is the edge-type-specific matrix.

- **Aggregation operation**. For aggregation operation, since the Attention() function's results have been normalized by softmax function, they simply use average function as Aggregation$(\cdot)$. Finally, they employ meta-path-specific projection followed by residual connection to learn the final representation of each nodes as follows:

$$\begin{aligned}
\mathbf{z}_i^{(k)} &= \sum_{v_j \in N(v_i)} (\text{Attention}^{(k)}(v_i, e_{i,j}, v_j)\text{Message}^{(k)}(v_i, e_{i,j}, v_j)), \\
\mathbf{h}_i^{(k)} &= \text{A_linear}_{\Phi_{v_i,e_{i,j}v_j}}(\sigma(\mathbf{z}_i^{(k)})) + \mathbf{h}^{(k-1)},
\end{aligned} \tag{77}$$

  where $\text{A\_linear}_{\Phi_{v_i,e_{i,j}v_j}}(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^d$ is the meta-relation-specific projection.

- **Relative Temporal Encoding** To tackle the graph's time dependency, they propose the Relative Temporal Encoding mechanism to each node's embedding. Technically, they calculate the timestamp difference of target and source nodes as $\delta_{i,j} = T(v_j) - T(v_i)$, where $T(\cdot)$ is the timestamp of the node. Then they project the time gap to the specific embedding space. This temporal encoding is added to the source nodes' representation before GNN encoding.

0.6 GNN Based Encoder-Decoder Models

Encoder-decoder architecture is one of the most widely used machine learning framework in the NLP field, such as the Sequence-to-Sequence (Seq2Seq) models(Sutskever *et al.*, 2014; Cho *et al.*, 2014). Given the great power of GNNs for modeling graph-structured data, very recently, many research efforts have been made to develop GNN-based encoder-decoder frameworks including Graph-to-Sequence (Song *et al.*, 2018c; Xu *et al.*, 2018b), Graph-to-Tree (Li *et al.*, 2020a; Zhang *et al.*, 2020b) and Graph-to-Graph (Guo *et al.*, 2019a; Shi *et al.*, 2020) models, as shown in Figure 13. In this section, we will first introduce the typical Seq2Seq models, and then discuss various graph-based encoder-decoder models for various NLP tasks.

0.6.1 Sequence-to-Sequence Models

Sequence-to-Sequence (Seq2Seq) learning (Sutskever *et al.*, 2014; Cho *et al.*, 2014) is one of the most widely used machine learning paradigms in the NLP field. In this section, we first give a brief overview of Seq2Seq learning and introduce some typical Seq2Seq techniques. Then we pinpoint some known limitations of Seq2Seq learning as well as its solutions, namely, incorporating more structured encoder-decoder models as alternatives to Seq2Seq models so as to encode more complex data structures.

Overview

Sequence-to-Sequence (Seq2Seq) models were originally developed by Sutskever *et al.* (2014) and Cho *et al.* (2014) for solving general sequence-to-sequence problems (e.g., machine translation). The Seq2Seq model is an end-to-end

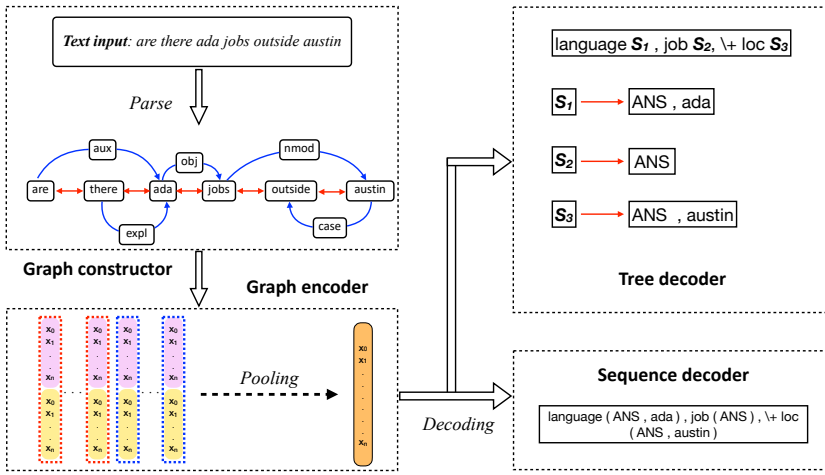


Figure 13: Overall architecture for graph based encoder-decoder model which contains both the Graph2Seq and Graph2Tree models. Input and output are from **JOBS640** dataset (Luke, 2005) . Nodes like $\boldsymbol{S}_1$, $\boldsymbol{S}_2$ stand for sub-tree nodes, which new branches are generated from.

encoder-decoder framework which learns to map a variable-length input sequence to a variable-length output sequence. Basically, the idea is to use an RNN-based encoder to read the input sequence (i.e., one token at a time), to build up a fixed-dimensional vector representation, and then use an RNN-based decoder to generate the output sequence (i.e., one token at a time) conditioned on the encoder output vector. The decoder is essentially a RNN language model except that it is conditioned on the input sequence. One of the most common Seq2Seq variants is to apply a Bidirectional LSTM encoder to encode the input sequence, and apply a LSTM decoder to decode the output sequence (Sutskever *et al.*, 2014). Other Seq2Seq variants replace LSTM with Gated Recurrent Units (GRUs) (Cho *et al.*, 2014), Convolutional Neural Networks (CNNs) (Gehring *et al.*, 2017) or Transformer models (Vaswani *et al.*, 2017).

Despite the promising achievements in many NLP applications such as machine translation, the original Seq2Seq models suffer a few issues such as the information bottleneck of the fixed-dimensional intermediate vector representation, and the exposure bias of cross-entropy based sequence training. In the original Seq2Seq architecture, the intermediate vector representation becomes an information bottleneck because it summarizes the rich information

of the input sequence as a fixed-dimensional embedding, which serves as the only knowledge source for the decoder to generate a high-quality output sequence. In order to increase the learning capacity of the original Seq2Seq models, many effective techniques have been proposed.

Approach

The attention mechanism (Bahdanau *et al.*, 2015; Luong *et al.*, 2015) was developed to learn the soft alignment between the input sequence and output sequence. Specifically, at each decoding step t, an attention vector indicating a probability distribution over the source words is computed as

$$\begin{aligned} e_i^t &= \mathrm{f}(\mathbf{h}_i, \mathbf{s}_t) \\ \mathbf{a}^t &= \mathrm{softmax}(\mathbf{e}^t), \end{aligned} \tag{78}$$

where f can be arbitrary neural network computing the relatedness between the decoder state $\mathbf{s}_t$ and encoder hidden state state $\mathbf{h}_i$. One common option is to apply an additive attention mechanism $\mathrm{f}(\mathbf{h}_i, \mathbf{s}_t) = \mathbf{v}^T \tanh(\mathbf{W}_h \mathbf{h}_i + \mathbf{W}_s \mathbf{s}_t + b)$ where $\mathbf{v}$, $\mathbf{W}_h$, $\mathbf{W}_s$ and b are learnable weights. Given the attention vector $\mathbf{a}^t$ at the t-th decoding step, the context vector can be computed as a weighted sum of the encoder hidden states, formulated as

$$\mathbf{h}_t^* = \sum_i a_i^t \mathbf{h}_i, \tag{79}$$

The computed context vector will be concatenated with the decoder state, and fed through some neural network for producing a vocabulary distribution.

The copying mechanism (Vinyals *et al.*, 2015; Gu *et al.*, 2016) was introduced to directly copy tokens from the input sequence to the output sequence in a learnable manner. This can be very helpful in some scenarios where the output sequence refers to some named entities or out-of-vocabulary tokens in the input sequence. Specifically, at each decoding step t, a generation probability will be calculated for deciding whether to generate a token from the vocabulary or copy a token from the input sequence by sampling from the attention distribution $\mathbf{a}^t$. The generation probability can be computed as

$$p_{\text{gen}} = \sigma(\mathbf{w}_{h^*}^T \mathbf{h}_t^* + \mathbf{w}_s^T \mathbf{s}_t + \mathbf{w}_x^T \mathbf{x}_t + b_{\text{ptr}})), \tag{80}$$

where $\mathbf{w}_{h^*}$, $\mathbf{w}_s$, $\mathbf{w}_x$ and b_{ptr} are learnable weights, σ is a sigmoid function, and p_{gen} is a scalar between 0 and 1.

The coverage mechanism (Tu *et al.*, 2016) was proposed to encourage the full utilization of different tokens in the input sequence. This can be useful in some NLP tasks such as machine translation. Specifically, at each decoding step t, a coverage vector $\mathbf{c}^t$ which is the aggregated attention vectors over all previous decoding steps will be computed as

$$\mathbf{c}^t = \sum_{t'=0}^{t-1} \mathbf{a}^{t'}. \tag{81}$$

In order to encourage better utilization of those source tokens that have not received enough attention scores so far, the above coverage vector will be used as extra input to the aforementioned attention mechanism Eq. (78), that is,

$$e_i^t = \mathrm{f}(\mathbf{h}_i, \mathbf{s}_t, c_i^t) \tag{82}$$

To avoid generating repetitive text, a coverage loss is calculated at each decoding step to penalize repeatedly attending to the same locations, formulated as,

$$\mathrm{covloss}_t = \sum_i \min(a_i^t, c_i^t) \tag{83}$$

The above coverage loss essentially penalizes the overlap between the attention vector and the coverage vector, and is bounded to $\sum_i a_i^t = 1$. It will be reweighted and added to the overall loss function.

The exposure bias occurs when during the training phase, the ground-truth token is used as the input (i.e., for better supervision) to the decoder for predicting the next token, while in the inference phase, the decoder's prediction from the previous time step is used as the input for next step prediction (due to no access to the ground-truth token). In order to reduce this gap between training and inference phases and thus increase the generalization ability of the original Seq2Seq models, scheduled sampling (Bengio *et al.*, 2015) was proposed to alleviate this issue by taking as input either the decoder's prediction from the previous time step or the ground truth with some probability for next step prediction, and gradually decreasing the probability of feeding in the ground truth at each iteration of training. The celebrated Seq2Seq models equipped with the above effective techniques have achieved great successes in a wide range of NLP applications such as neural machine translation (Bahdanau *et al.*, 2015; Luong *et al.*, 2015; Gehring *et al.*, 2017), text summarization (Nallapati *et al.*, 2016; See *et al.*, 2017; Paulus *et al.*, 2018), text generation (Song *et al.*,

2017), speech recognition (Zhang *et al.*, 2017a), and dialog systems (Serban *et al.*, 2016; Serban *et al.*, 2017).

Discussions

Seq2Seq models were originally developed to solve sequence-to-sequence problems, that is, to map a sequential input to a sequential output. However, many NLP applications naturally admit graph-structured input data such as dependency graphs (Fu *et al.*, 2019; Chen *et al.*, 2020i), constituency graphs (Li *et al.*, 2020a; Marcheggiani and Titov, 2020), AMR graphs (Beck *et al.*, 2018a; Song *et al.*, 2019), IE graphs (Cao *et al.*, 2019c; Huang *et al.*, 2020b) and knowledge graphs (Nathani *et al.*, 2019a; Wu *et al.*, 2019a). In comparison with sequential data, graph-structured data is able to encode rich syntactic or semantic relationships among objects. Moreover, even if the raw input is originally represented in a sequential form, it can still benefit from explicitly incorporating rich structural information (e.g., domain-specific knowledge) to the sequence. The above situations essentially call for an encoder-decoder framework for learning a graph-to-X mapping where X can stand for a sequence, tree or even graph. Existing Seq2Seq models face a significant challenge in learning an accurate mapping from graph to the appropriate target due to its incapability of modeling complex graph-structured data.

Various attempts have been made in order to extend Seq2Seq models to handle Graph-to-Sequence problems where the input is graph-structured data. A simple and straightforward approach is to directly linearize the structured graph data into the sequential data (Iyer *et al.*, 2016; Gómez-Bombarelli *et al.*, 2018; Liu *et al.*, 2017), and apply the Seq2Seq models to the resulting sequence. However, this kind of approaches suffer significant information loss, which leads to downgraded performance. The root cause of RNN's incapability of modeling complex structured data is because it is a linear chain. In light of this, some research efforts have been devoted to extend Seq2Seq models. For instance, Tree2Seq (Eriguchi *et al.*, 2016) extends Seq2Seq models by adopting Tree-LSTM (Tai *et al.*, 2015) which is a generalization of chain-structured LSTM to tree-structured network topologies. Set2Seq (Vinyals *et al.*, 2016) is an extension of Seq2Seq models that goes beyond sequences and handles the input set using the attention mechanism. Although these Seq2Seq

extensions achieve promising results on certain classes of problems, none of them can model arbitrary graph-structured data in a principled way.

0.6.2 Graph-to-Sequence Models

Overview

To address the aforementioned limitations of Seq2Seq models on encoding rich and complex data structures, recently, a number of graph-to-sequence encoder-decoder models for NLP tasks have been proposed (Bastings *et al.*, 2017; Beck *et al.*, 2018a; Song *et al.*, 2018c; Xu *et al.*, 2018b). This kind of Graph2Seq models typically adopt a GNN based encoder and a RNN/Transformer based decoder. Compared to the Seq2Seq paradigm, the Graph2Seq paradigm is better at capturing the rich structure information of the input text and can be applied to arbitrary graph-structured data. Graph2Seq models have shown superior performance in comparison with Seq2Seq models in a wide range of NLP tasks including neural machine translation (Bastings *et al.*, 2017; Marcheggiani *et al.*, 2018; Beck *et al.*, 2018a; Song *et al.*, 2019; Xu *et al.*, 2020c; Yao *et al.*, 2020; Yin *et al.*, 2020; Cai and Lam, 2020c), AMR-to-text (Beck *et al.*, 2018a; Song *et al.*, 2018c; Damonte and Cohen, 2019; Ribeiro *et al.*, 2019a; Zhu *et al.*, 2019b; Wang *et al.*, 2020g; Guo *et al.*, 2019c; Yao *et al.*, 2020; Wang *et al.*, 2020f; Cai and Lam, 2020c; Bai *et al.*, 2020; Song *et al.*, 2020; Zhao *et al.*, 2020b; Zhang *et al.*, 2020d; Jin and Gildea, 2020), text summarization (Fernandes *et al.*, 2019; Xu *et al.*, 2020a; Huang *et al.*, 2020b; Zhang *et al.*, 2020c), question generation (Chen *et al.*, 2020i; Wang *et al.*, 2020d), KG-to-text (Koncel-Kedziorski *et al.*, 2019), SQL-to-text (Xu *et al.*, 2018a), code summarization (Liu *et al.*, 2021b), and semantic parsing (Xu *et al.*, 2018c).

Approach

Most proposed Graph2Seq models were designed for tackling particular NLG tasks. In the followings, we will discuss some common techniques adopted in a wide rage of Graph2Seq variants, which include both graph-based encoding techniques and sequential decoding techniques.

Graph-based Encoders Early Graph2Seq methods and their follow-up works (Bastings *et al.*, 2017; Marcheggiani *et al.*, 2018; Damonte and Cohen, 2019; Guo *et al.*, 2019c; Xu *et al.*, 2020a; Xu *et al.*, 2020c; Zhang *et al.*, 2020c; Zhang *et al.*, 2020d) mainly used some typical GNN variants as the graph encoder incldluing GCN, GGNN, GraphSAGE and GAT. Since the edge direction in a NLP graph often encodes critical information about the semantic relations between two vertices, it is often extremely helpful to capture the bidirectional information of text (Devlin *et al.*, 2019). In the literature of Graph2Seq paradigm, some efforts have been made to extend the existing GNN models to handle directed graphs. The most common strategy is to introduce separate model parameters for different edge directions (i.e., incoming/outgoing/self-loop edges) when performing neighborhood aggregation (Marcheggiani *et al.*, 2018; Song *et al.*, 2018c; Song *et al.*, 2019; Xu *et al.*, 2020c; Yao *et al.*, 2020; Wang *et al.*, 2020g; Guo *et al.*, 2019c).

Besides the edge direction information, many graphs in NLP applications are actually multi-relational graphs where the edge type information is very important for the downstream task. In order to encode edge type information, some works (Simonovsky and Komodakis, 2017; Chen *et al.*, 2018b; Ghosal *et al.*, 2020; Wang *et al.*, 2020b; Schlichtkrull *et al.*, 2018; Teru *et al.*, 2020) have extended them by having separate model parameters for different edge types (i.e., similar ideas have been used for encoding edge directions). However, in many NLP applications (e.g., KG-related tasks), the total number of edge types is large, hence the above strategy can have severe scalability issues. To this end, some works (Beck *et al.*, 2018a; Koncel-Kedziorski *et al.*, 2019; Yao *et al.*, 2020; Ribeiro *et al.*, 2019a; Guo *et al.*, 2019c; Chen *et al.*, 2020g) proposed to bypass this problem by converting a multi-relational graph to a Levi graph (Levi, 1942) and then utilize existing GNNs designed for homogeneous graphs as encoders. Another commonly adopted technique is to explicitly incorporate edge embeddings into the message passing mechanism (Marcheggiani *et al.*, 2018; Song *et al.*, 2018c; Song *et al.*, 2019; Zhu *et al.*, 2019b; Wang *et al.*, 2020f; Cai and Lam, 2020c; Wang *et al.*, 2020g; Song *et al.*, 2020; Liu *et al.*, 2021b; Jin and Gildea, 2020).

Besides the above widely used GNN variants, some Graph2Seq works also explored other GNN variants such as GRN (Song *et al.*, 2018c; Song *et al.*, 2019) and GIN (Ribeiro *et al.*, 2019a). Notably, GRN is also capable of handling multi-relational graphs by explicitly including edge embeddings in

the LSTM-style message passing mechanism.

Node & Edge Embeddings Initialization For GNN based approaches, initialization of nodes and edges is extremely critical. While both CNNs and RNNs are good at capturing local dependencies among consecutive words in text, GNNs do well in capturing local dependencies among neighboring nodes in a graph. Many works on Graph2Seq have shown benefits of initializing node and/or edge embeddings by applying CNNs (Bastings *et al.*, 2017; Marcheggiani *et al.*, 2018) or bidirectional RNNs (BiRNNs) (Bastings *et al.*, 2017; Marcheggiani *et al.*, 2018; Fernandes *et al.*, 2019; Xu *et al.*, 2018b; Xu *et al.*, 2018a; Koncel-Kedziorski *et al.*, 2019; Cai and Lam, 2020c; Wang *et al.*, 2020d; Chen *et al.*, 2020i; Liu *et al.*, 2021b) to the word embedding sequence before applying the GNN based encoder. Some works also explored to initialize node/edge embeddings with BERT embeddings+BiRNNs (Xu *et al.*, 2020a; Chen *et al.*, 2020i) or RoBERTa+BiRNNs (Huang *et al.*, 2020b).

Sequential Decoding Techniques Since the main difference between Seq2Seq and Graph2Seq models is on the encoder side, common decoding techniques used in Seq2Seq models such as attention mechanism (Bahdanau *et al.*, 2015; Luong *et al.*, 2015), copying mechanism (Vinyals *et al.*, 2015; Gu *et al.*, 2016), coverage mechanism (Tu *et al.*, 2016), and scheduled sampling (Bengio *et al.*, 2015) can also be adopted in Graph2Seq models with potential modifications.

Some efforts have been made to adapt common decoding techniques to the Graph2Seq paradigm. For example, in order to copy the whole node attribute containing multi-token sequence from the input graph to the output sequence, Chen *et al.* (2020g) extended the token-level copying mechanism to the node-level copying mechanism. To combine the benefits of both sequential encoder and graph encoder, Pan *et al.* (2020) and Sachan *et al.* (2020) proposed to fuse their outputs to a single vector before feeding it to a decoder. Huang *et al.* (2020b) designed separate attention modules for sequential encoder and graph

encoder, respectively.

$$\begin{aligned} a_i^v &= \text{attn\_v}(\mathbf{s}_t, \mathbf{h}_i^v) \\ \mathbf{c}^v &= \sum_i a_i^v \mathbf{h}_i^v \\ a_j^s &= \text{attn\_s}(\mathbf{s}_t, \mathbf{h}_j^s, \mathbf{c}^v) \\ \mathbf{c}^s &= \sum_i a_j^s \mathbf{h}_j^s \\ \mathbf{c} &= \mathbf{c}^v || \mathbf{c}^s \end{aligned} \tag{84}$$

where $\mathbf{h}_i^v$ and $\mathbf{h}_j^s$ are the graph encoder outputs and sequential encoder outputs, respectively. $\mathbf{c}$ is the concatenation of the graph context vector $\mathbf{c}^v$ and sequential context vector $\mathbf{c}^s$.

In order to tackle the limitations (e.g., exposure bias and discrepancy between the training and inference phases) of cross-entropy based sequential training, Chen *et al.* (2020i) proposed to train the Graph2Seq system by minimizing a hybrid loss combining both cross-entropy loss and reinforcement learning (Williams, 1992) loss. While LSTM or GRU based decoders are the most commonly used decoder in Graph2Seq models, some works also employed a Transformer based decoder (Koncel-Kedziorski *et al.*, 2019; Zhu *et al.*, 2019b; Yin *et al.*, 2020; Wang *et al.*, 2020f; Cai and Lam, 2020c; Bai *et al.*, 2020; Wang *et al.*, 2020g; Song *et al.*, 2020; Zhao *et al.*, 2020b; Jin and Gildea, 2020).

Discussions

There are some connections and differences between Graph2Seq models and Transformer-based Seq2Seq models, and many of them have already been discussed above when we talk about the connections and differences between GNNs and Transformer models. It is worth noting that there is a recent trend in combining the benefits of the both paradigms, thus making them less distinguishable. Many recent works designed various graph transformer based generation models (as we discussed above) which employ a graph-based encoder combining both the benefits of GNNs and Transformer, and a RNN/Transformer based decoder.

Despite the great success of Graph2Seq models, there are some open challenges. Many of these challenges are essentially the common challenges of applying GNNs for graph representation learning, including how to better

model multi-relational or heterogeneous graphs, how to scale to large-scale graphs such as knowledge graphs, how to conduct joint graph structure and representation learning, how to tackle the over-smoothing issue and so on. In addition, Graph2Seq models also inherit many challenges that Seq2Seq models have, e.g., how to tackle the limitations of cross-entropy based sequential training (e.g., exposure bias and discrepancy between the training and inference phases).

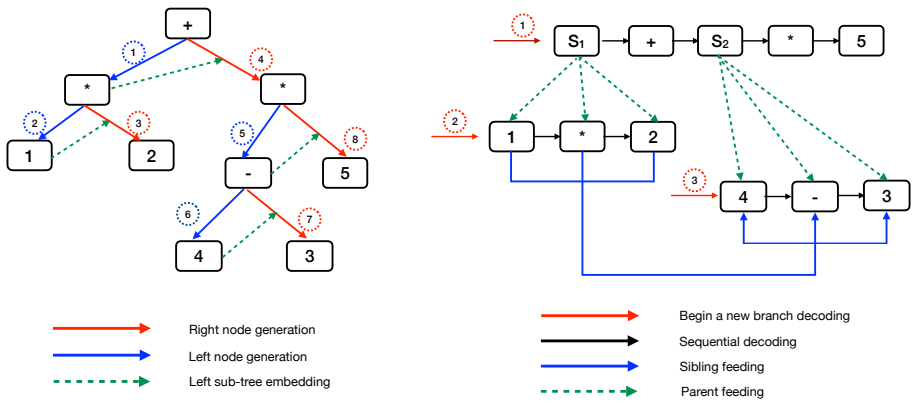


Figure 14: Equation: (1 * 2) + (4 - 3) * 5. Left: a DFS-based tree decoder example, the number stands for the order of the decoding actions. Right: a BFS based tree decoder example. Nodes like S_1, S_2 stand for sub-tree nodes, and once a sub-tree node generated, decoder will start a new branch for a new descendant decoding process. The number stands for the order of different branching decoding processes.

0.6.3 Graph-to-Tree Models

Overview

Compared to Graph2Seq model, which considers the structural information in the input side, many NLP tasks also contain outputs represented in a complex structured, such as trees, which are also rich in structural information at the output side, e.g., syntactic parsing(Ji *et al.*, 2019)(Yang and Deng, 2020), semantic parsing(Li *et al.*, 2020a)(Xu *et al.*, 2018c), math word problem solving(Li *et al.*, 2020a)(Zhang *et al.*, 2020b). It is a natural choice for us to consider the structural information of these outputs. To this end, some Graph2Tree models are proposed to incorporate the structural information

in both the input and output side, which make the information flow in the encoding-decoding process more complete.

Approach

To illustrate how the Graph2Tree model works, we will introduce how different components of the Graph2Tree model operate here, including: graph construction, graph encoder, attention mechanism, and tree decoder.

Graph construction The graph construction module, which is usually highly related to specific tasks, could be divided into two categories: one with auxiliary information and one without auxiliary information. For the former, Li *et al.* (2020a) use syntactic graph in both semantic parsing and math word problem solving tasks, which consists of the original sentence and the syntactic pasing tree (dependency and constituency tree). And the input graph in (Yin *et al.*, 2018) considers the graph composed of the abstract syntax tree (AST) of a fragment of source code. For the latter, the input can usually form a task graph itself without auxiliary information. For example, Zhang *et al.* (2020b) employ the relationship between different numbers in the math word problem in the graph construction module.

Graph encoder Graph encoder is used for embedding the input graph into the latent representation. To implementing the graph encoder module, several graph2tree models use relatively simple GNN models, such as GCN(Kipf and Welling, 2016), GGNN(Li *et al.*, 2015), and GraphSAGE(Hamilton *et al.*, 2017a). For (Li *et al.*, 2020a), it uses a bidirectional variant of the GraphSage model, and Zhang *et al.* (2020b) exploit the GCN model before a transformer layer. And Yin *et al.* (2018) simply adopt the GGNN model as its neural encoder.

Attention The attention module is a key component in an encoder-decoder framework, which carry the important information for bridging the input and output semantics. In the graph2tree model, the input graph often contains different types of nodes(Li *et al.*, 2020a)(Zhang *et al.*, 2020b), while the traditional attention module can not distinguish between these nodes. In (Li *et al.*, 2020a), the author uses the separate attention module to calculate the

attention vector for different nodes in the input graph where some nodes is from the original sentence, and others are composed of the nodes in parsing trees generated by the external parser. It has been validated that distinguishing these two types of nodes could facilitate better learning process than the original attention module. This idea is similar to the application of Tree2Seq(Eriguchi *et al.*, 2016) attention module in machine translation.

Specifically in (Li *et al.*, 2020a), the decoder generates the tree structure by representing some branching nodes as non-terminal nodes, i.e., node S_1 in Figure 14. Once these nodes generated, the decoder will start a new sequential decoding process. The decoder hidden state $\mathbf{s}_t$ at time step t is calculated a

$$\mathbf{s}_t = f_{decoder}(y_{t-1}, \mathbf{s}_{t-1}; \mathbf{s}_{par}; \mathbf{s}_{sib}), \tag{85}$$

where the $\mathbf{s}_{par}, \mathbf{s}_{sib}$ stand for the parent node hidden state and sibling node hidden state as illustrated in Figure 14. After the current hidden state generated, the output module including attention layer is calculated as follows:

$$\alpha_{t(v)} = \frac{\exp(score(\mathbf{z}_v, \mathbf{s}_t))}{\exp(\sum_{k=1}^{V_1} score(\mathbf{z}_k, \mathbf{s}_t))}, \forall v \in \mathcal{V}_1 \tag{86}$$

$$\beta_{t(v)} = \frac{\exp(score(\mathbf{z}_v, \mathbf{s}_t))}{\exp(\sum_{k=1}^{V_2} score(\mathbf{z}_k, \mathbf{s}_t))}, \forall v \in \mathcal{V}_2 \tag{87}$$

$$\mathbf{c}_{v_1} = \sum \alpha_{t(v)} \mathbf{z}_v, \forall v \in \mathcal{V}_1 \tag{88}$$

$$\mathbf{c}_{v_2} = \sum \beta_{t(v)} \mathbf{z}_v, \forall v \in \mathcal{V}_2 \tag{89}$$

where $\mathbf{z}_v$ denotes to the learned node embedding for node $\mathbf{v}$, $\mathcal{V}_1$ denotes to the node set including all words from original sentences, and $\mathcal{V}_2$ denotes to another node set including all other nodes. We then concatenate the context vector $\mathbf{c}_{v_1}$, context vector $\mathbf{c}_{v_2}$ and decoder hidden state $\mathbf{s}_t$ to compute the final attention hidden state at this time step as:

$$\tilde{\mathbf{s}_t} = \tanh(W_c \cdot [\mathbf{c}_{v_1}; \mathbf{c}_{v_2}; \mathbf{s}_t] + b_c), \tag{90}$$

where W_c and b_c are learnable parameters. The final context vector $\tilde{\boldsymbol{s_t}}$ is further fed to the output layer which is a softmax function after a feed-forward layer.

Tree decoder The output of some applications (i.e., semantic parsing, code generation, and math word problem) contain structural information, for example, the output in math word problem is a mathematical equation, which can be expressed naturally by the data structure of the tree. To generate these kinds of outputs, tree decoders are widely used in these tasks. Tree decoders can be divided into two main parts as shown in Figure 14, namely, dfs (depth first search) based tree decoder, and bfs (breadth first search) based tree decoder.

For bfs-based decoders(Li *et al.*, 2020a; Dong and Lapata, 2016; Alvarez-Melis and Jaakkola, 2016), the main idea is to represent all the sub-trees in the tree as non-terminal nodes, and then use sequence decoding to generate intermediate results. If the results contains non-terminals, then we start branching (begin a new decoding process) with this node as the new root node, until the entire tree is expanded.

For dfs-based decoders(Zhang *et al.*, 2020b; Yin *et al.*, 2018), they regards the entire tree generation process as a sequence of actions. For example, in the generation of a binary tree (mathematical equation) in (Zhang *et al.*, 2020b), the root node is generated in priority at each step, following by the generation of the left child node. After all the left child nodes are generated, a bottom-up manner is adopted to begin the generation of the right child nodes.

In addition, the tree decoder is constantly evolving, and some techniques are proposed to collect more information during the decoding process or leverage the information from the input or output, such as parent feeding(Dong and Lapata, 2016), sibling feeding(Li *et al.*, 2020a), sub-tree copy(Yin *et al.*, 2018), tree based re-ranking(Do and Rehbein, 2020) and other techniques. At the same time, the wide application of the transformer model also brings about many transformer based tree decoders(Sun *et al.*, 2020c)(Li *et al.*, 2020c), which proves the wide application of tree decoder and Graph2tree model.

0.6.4 Graph-to-Graph Models

The graph-to-graph models are typically utilized for solving graph transformation problem as a graph encoder-decoder model. The graph encoder generates the latent representation of each node in the graph or generate one graph-level latent representation for the whole graph via the GNNs. The graph decoder then generates the output target graphs based on the node-level or graph-level

latent representations from the encoder. In this section, we first introduce graph-to-graph transformation problem and the typical NLP applications that can be formalized as graph-to-graph transformation problems. Then, we introduce the specific techniques for a Graph-to-graph model for information extraction.

Overview

Graph-to-graph transformation Graph-to-graph models aims to deal with the problem of deep graph transformation (Guo *et al.*, 2018). The goal of graph transformation is to transform an input graph in the source domain to the corresponding output graphs in the target domain via deep learning. Emerging as a new while important problem, deep graph transformation has multiple applications in many areas, such as molecule optimization (Shi *et al.*, 2020; Zhou *et al.*, 2020a; Do *et al.*, 2019) and malware confinement in cyber security (Guo *et al.*, 2019a). Considering the entities that are being transformed during the translation process, there are three categories of sub-problems: node transformation, edge transformation, and node-edge-co-transformation. For node transformation, only the node set or nodes' attributes in the input graph can change during the transformation process. For edge transformation, only the graph topology or edge' attributes in the input graph can change during the transformation process. While for node-edge-co-transformation, both the attributes of nodes and edges can change.

Graph-to-Graph for NLP Since the natural language or information knowledge graphs can be naturally formalized as graphs with a set of nodes and their relationships, many generation tasks in the domain of NLP can be formalized as a graph transformation problem, which can further be solved by the graph-to-graph models. In this way, the semantic structural information of both the input and output sentences can be fully utilized and captured. Here, two important NLP tasks (i.e., information extraction and semantic parsing), which can be formalized as the graph-to-graph problems, are introduced as follows.

Graph Transformation for Information Extraction. Information extraction is to extract the structured information from a text, which usually consists of name entity recognition, relation extraction and co-reference linking. The

problem of information extraction can be formalized as a graph transformation problem, where the input is the dependency or constituency graph of a text and the output is the information graph. In input dependency or constituency graph, each node represents a word token and each edge represent the dependency relationship between two nodes. In output information graph, each node represents a name entity and each edge represents either the semantic relation or the co-reference link between two entities. In this way, the information extraction is about generating the output information graph given the input dependency or constituency graph.

Graph Transformation for Semantic Parsing. The task of semantic parsing is about mapping natural language to machine interpretable meaning representations, which in turn can be expressed in many different formalisms, including lambda calculus, dependency-based compositional semantics, frame semantics, abstract meaning representations (AMR), minimal recursion semantics, and discourse representation theory (Fancellu *et al.*, 2019). Explicitly or implicitly, a representation in any of these formalisms can be expressed as a directed acyclic graph (DAG). Thus, semantic parsing can also be formalized as a graph transformation problem, where the input is the dependency or constituency graph and the output is the directed acyclic graph for semantics. For example, the semantic formalism for AMR can be encoded as a rooted, directed, acyclic graph, where nodes represent concepts, and labeled directed edges represent the relationships between them (Flanigan *et al.*, 2014; Fu *et al.*, 2021).

Sequence-to-graph transformation can be regarded as a special case of the graph-to-graph, where the input sequence is a line-graph. Sequence-to-graph models are popularly utilized for AMR parsing tasks, where the goal is to learning the mapping from a sentence to its AMR graph (Zhang *et al.*, 2019e). To generate the AMR tree with indexed node, the approach to parsing is formalized as a two-stage process: node prediction and edge prediction. The whole process is implemented by an pointer-network, where the encoder is a multi-layer bi-direction-RNN and the nodes in the target graphs are predicted in sequence. After this, the edges among each pair of nodes are predicted based on the learnt embedding of the ending nodes.

Approach

In this subsection, we introduce an example graph-to-graph model in dealing with the task of information extraction by describing its challenges and methodologies.

Challenges for Graph-to-Graph IE. There are three main challenges in solving the graph-to-graph IE problem: (1) Different resolution between the input and output graphs. The nodes in the input dependency graph represent word tokens, while the nodes in the output information graph represent name entities; (2) Difficult to involve both the sentence-level and word-level. To learn the word embedding in the graph encoder, it is important to consider both the word interactions in a sentence and the sentence interactions in a text; and (3) Difficult to model the dependency between entity relations and co-reference links in the graph decoder. The generation process of entity relation and co-reference links are dependent on each other. For example, if words "Tom" and "He" in two separate sentences have a co-reference link, and "Tom" and "London" has the relation of "born_In", then "He" and "London" should also have the relation of "born_In".

Methodology. To solve the above mentioned challenges for the graph-to-graph IE task, here we introduce an end-to-end encoder-decoder based graph-to-graph transformation model, which transforms the input constructed graphs of text into the information graphs which contains name entities as well as co-reference links and relations among entities. The whole model consists of a hierarchy graph encoder for span embedding and a parallel decoder with co-reference link and entity relation generation.

First, to construct the initial graphs, the dependency information are formalized into a heterogeneous graph which consists of nodes representing word tokens (i.e., word-level nodes) and nodes representing sentences (i.e., sentence-level nodes). There are also three types of edges in the graph. One type of edges represent the dependency relationships between word-level nodes (i.e., dependency edges). One type of edges represent the adjacent relationship between sentence-level nodes (i.e., adjacent edges). The last type of edges represent the belongingness between the word-level and sentence-level nodes (i.e., interactive edges).

Second, the constructed heterogeneous graph is inputted into the encoder, which is based on a hierarchy graph convolution neural network. Specifi-

cally, for each layer, the conventional message passing operations are first conducted along the dependency and adjacent edges to update the embedding of word-level and sentence-level nodes. Then, the conventional message passing operations are conducted along the interactive edges based on the newly updated word-level and sentence-level nodes' embedding. After several layers of propagation, the embedding of word-evel nodes will contains both the dependency and sentence-level information.

Third, based on the words' embedding from the encoder, the name entities can be first extracted via BIO Tagging (Marquez *et al.*, 2005). Thus, the entity-level embedding are then constructed by summing all of the embedding of the words it contains. Given the entity embedding, to model the dependency between the co-reference links and relations between entities, a parallel graph decoder (Guo *et al.*, 2018) that involves both co-reference link and entity relation generation processes is utilized. Specifically, given the entity embedding h_i and h_j of a pair of name entities v_i and v_j, the initial generated latent representation of co-reference link $\mathbf{c}_{i,j}^0$ is computed as:

$$\mathbf{c}_{i,j}^0 = \sum\nolimits_{m=1}^{C} (\sigma(h_i^m \bar{\mu}_j) + \sigma(h_j^m \bar{\nu}_i)), \tag{91}$$

where $\sigma(h_i^m \bar{\mu}_j)$ means the deconvolution contribution of node v_i to its edge representations with node v_j, which is made by the m-th entry of its node representations, and $\bar{\mu}_j$ represents one entry of the deconvolution filter vector $\bar{\mu} \in \mathbb{R}^{N \times 1}$ that is related to node v_j. The initial relation latent representation $e_{i,j}^0$ between a pair of name entities v_i and v_j can also be computed in the same way. C refers to the length of name entity embedding.

Given the initial latent representation of co-reference links and relations, the co-reference link representation $\mathbf{c}_{i,j}^l$ at the l-th layer is computed as follows:

$$\mathbf{c}_{i,j}^l = \sigma(\bar{\phi}_j^{l-1} \sum\nolimits_{k_1=1}^{N} [\mathbf{c}^{l-1}; \mathbf{e}^{l-1}]_{i,k_1}^{l-1} x_{k_1}) + \sigma(\bar{\psi}_i^l \sum\nolimits_{k_2=1}^{N} [\mathbf{c}^{l-1}; \mathbf{e}^{l-1}]_{k_2,j}^{l-1} x_{k_1}), \tag{92}$$

where $\bar{\phi}_j^{l-1} \sum_{k_1=1}^{N} [\mathbf{c}^{l-1}; \mathbf{e}^{l-1}]_{i,k_1}^{l-1} x_{k_1}$ can be interpreted as the decoded contribution of node v_i to its edge representations with node v_j, and $\bar{\phi}_j^{l-1}$ refers to the element of deconvolution filter vector that is related to node v_j. The output of the last "edge" deconvolution layer denotes the probability of the existence of an edge in the target graph. All the symbols σ refers to the activation functions. $[\mathbf{c}^{l-1}; \mathbf{e}^{l-1}]$ refers to the concatenation of all the co-reference and relation representations at $(l-1)$-th layer.

0.7 Applications

In this chapter, we will discuss a large variety of typical NLP applications using GNNs, including natural language generation, machine reading comprehension, question answering, dialog system, text classification, text matching, topic modeling, sentiment classification, knowledge graph, information extraction, semantic and syntactic parsing, reasoning, and semantic role labelling. We also provide the summary of all the applications with their sub-tasks and evaluation metrics in Table 3.

0.7.1 Natural Language Generation

Natural language generation (NLG) aims to generate high-quality, coherent and understandable natural languages given various form of inputs like text, speech and etc while we only focus on the linguistic form. Modern natural language generation methods usually take the form of encoder-decoder, which encodes the input sequences into latent space and predicts a collection of words based on the the latent representation. Most modern NLG pipelines can be divided into two steps: Encoding and Decoding, which are processed by two module: encoder and decoder. In this section, we provide a comprehensive overview of the auto-regressive graph-based methodologies which exploit graph structures in encoder in this thriving area covering 1) neural machine translation, 2) summarization, 3) question generation, 4) structural-data to text.

Neural Machine Translation

Background and Motivation The classic neural machine translation (NMT) system aims to map the source language's sentences into the target language without changing the semantic meaning. Most prior works (Bahdanau *et al.*, 2015; Luong *et al.*, 2015) adopt the attention-based sequence-to-sequence learning diagram, especially the RNN-based language model. Compared with the traditional machine translation models, these methods can produce much better performance without specific linguistic knowledge. However, these methods suffer from the long-dependency problem. With the development of attention mechanism, fully-attention-based models such as Transformer (Vaswani *et al.*, 2017), which captures the implicit correlations

Table 3: Typical NLP applications and relevant works using GNNs

Application	Task	Evaluation	References
NLG	Neural Machine Translation	BLEU	Bastings *et al.* (2017), Beck *et al.* (2018b), and Cai and Lam (2020b) Guo *et al.* (2019c), Marcheggiani *et al.* (2018), and Shaw *et al.* (2018) Song *et al.* (2019), Xiao *et al.* (2019), Xu *et al.* (2020c), and Yin *et al.* (2020)
	Summarization	ROUGE	Xu *et al.* (2020a), Wang *et al.* (2019e), and Li *et al.* (2020b) Fernandes *et al.* (2019) and Wang *et al.* (2020a) Cui *et al.* (2020b), Jia *et al.* (2020), and Zhao *et al.* (2020a) Jin *et al.* (2020a), Yasunaga *et al.* (2017), and LeClair *et al.* (2020)
	Structural-data to Text	BLEU, METEOR	Bai *et al.* (2020), Jin and Gildea (2020), and Xu *et al.* (2018a) Beck *et al.* (2018b), Cai and Lam (2020a), and Zhu *et al.* (2019c) Cai and Lam (2020b), Ribeiro *et al.* (2019a), and Song *et al.* (2020) Wang *et al.* (2020f), Yao *et al.* (2018), and Zhang *et al.* (2020d)
	Natural Question Generation	BLEU, METEOR, ROUGE	Chen *et al.* (2020g), Liu *et al.* (2019b), and Pan *et al.* (2020) Wang *et al.* (2020c), Sachan *et al.* (2020), and Su *et al.* (2020)
MRC and QA	Machine Reading Comprehension	F1, Exact Match	De Cao *et al.* (2018), Cao *et al.* (2019c), and Chen *et al.* (2020h) Qiu *et al.* (2019), Schlichtkrull *et al.* (2018), and Tang *et al.* (2020c) Tu *et al.* (2019a) and Song *et al.* (2018b) Fang *et al.* (2020a) and Zheng and Kordjamshidi (2020)
	Knowledge Base Question Answering	F1, Accuracy	Feng *et al.* (2020a) and Sorokin and Gurevych (2018b) Santoro *et al.* (2017) and Yasunaga *et al.* (2021)
	Open-domain Question Answering	Hits@1, F1	Han *et al.* (2020), Sun *et al.* (2019b), and Sun *et al.* (2018a)
	Community Question Answering	nDCG, Precision	Hu *et al.* (2019b) and Hu *et al.* (2020b)
Dialog Systems	Dialog State Tracking	Accuracy	Chen *et al.* (2018b) and Chen *et al.* (2020b)
	Dialog Response Generation	BLEU, METEOR, ROUGE	Hu *et al.* (2019e) and Bai *et al.* (2021)
	Next Utterance Selection	Recall@K	Liu *et al.* (2021c)
Text Classification		Accuracy	Chen *et al.* (2020f), Defferrard *et al.* (2016), and Henaff *et al.* (2015) Huang *et al.* (2019), Hu *et al.* (2020c), and Liu *et al.* (2020)
Text Matching		Accuracy, F1	Chen *et al.* (2017c) and Liu *et al.* (2019a)
Topic Modeling		Topic Coherence Score	Long *et al.* (2020) and Yang *et al.* (2020) Zhou *et al.* (2020a) and Zhu *et al.* (2018)
Sentiment Classification		Accuracy, F1	Zhang and Qian (2020) and Pouran Ben Veyseh *et al.* (2020) Chen *et al.* (2020e) and Tang *et al.* (2020a) Sun *et al.* (2019c), Wang *et al.* (2020b), and Zhang *et al.* (2019a) Ghosal *et al.* (2020), Huang and Carley (2019), and Chen *et al.* (2020a)
Knowledge Graph	Knowledge Graph Completion	Hits@N	Malaviya *et al.* (2020), Nathani *et al.* (2019a), and Teru *et al.* (2020) Bansal *et al.* (2019), Schlichtkrull *et al.* (2018), and Shang *et al.* (2019) Wang *et al.* (2019a), Wang *et al.* (2019g), and Zhang *et al.* (2020g)
	Knowledge Graph Alignment		Cao *et al.* (2019b), Li *et al.* (2019), and Sun *et al.* (2020b) Wang *et al.* (2018), Wang *et al.* (2020h), and Ye *et al.* (2019) Xu *et al.* (2019a) and Wu *et al.* (2019a)
Information Extraction	Named Entity Recognition	Precision, Recall, F1	Luo and Zhao (2020), Ding *et al.* (2019b), and Gui *et al.* (2019) Jin *et al.* (2019) and Sui *et al.* (2019)
	Relation Extraction		Qu *et al.* (2020), Zeng *et al.* (2020), and Sahu *et al.* (2019) Guo *et al.* (2019b) and Zhu *et al.* (2019a)
	Joint Learning Models		Fu *et al.* (2019), Luan *et al.* (2019), and Sun *et al.* (2019a)
Parsing	Syntax-related	Accuracy	Do and Rehbein (2020), Ji *et al.* (2019), and Yang and Deng (2020)
	Semantics-related		Bai *et al.* (2020) and Zhou *et al.* (2020b) Shao *et al.* (2020), Bogin *et al.* (2019a), and Bogin *et al.* (2019b)
Reasoning	Math Word Problem Solving	Accuracy	Li *et al.* (2020a), Lee *et al.* (2020), and Wu *et al.* (2020b) Zhang *et al.* (2020b) and Ferreira and Freitas (2020)
	Natural Language Inference		Kapanipathi *et al.* (2020) and Wang *et al.* (2019f)
	Commonsense Reasoning		Zhou *et al.* (2018a), Lin *et al.* (2019b), and Lin *et al.* (2019a)
Semantic Role Labelling		Precision, Recall, F1	Marcheggiani and Titov (2020), Xia *et al.* (2020), and Zhang *et al.* (2020a) Li *et al.* (2018c), Marcheggiani and Titov (2017), and Fei *et al.* (2020)

by self-attention, have made a breakthrough and achieved a new state-of-art. Although these works achieve great success, they rarely take the structural information into account, such as the syntactic structure. Recently, with the help of powerful GNNs, many researchers further boost the performance by mining the structural knowledge contained in the unstructured texts.

Methodologies Most GNN-based NMT methods cast the conventional seq2seq diagram to the Graph2Seq architecture. They firstly convert the input texts to graph-structured data, and then employ the GNN-based encoder to exploit the structural information. In this section, we introduce and summarize some representative GNN-related techniques adopted in recent NMT approaches regarding graph construction and representation learning.

- **Graph Construction**. Various static graphs have been introduced to the NMT task to tackle corresponding challenges. Bastings *et al.* (2017), Beck *et al.* (2018b), Cai and Lam (2020b), and Guo *et al.* (2019c) first converted the given texts into syntactic dependency graph. Such structure doesn't take semantic relations of words into account. Intuitively, it is beneficial to represent the redundant sentences by high-level semantic structure abstractions. To this end, Marcheggiani *et al.* (2018) construct the semantic-role-labeling based dependency graph for the given texts. What's more, Beck *et al.* (2018b) and Song *et al.* (2019) construct the AMR graph for the sentences which can cover more semantic correlations. Besides the classic graph types, some specifically designed graphs (app-driven graphs) are proposed to address the unique challenges. Although source sentences in NMT are determined, either word-level or subword-level segmentations have multiple choices to split a source sequence with different word segments or different subword vocabulary sizes. Such a phenomenon is proposed to affect the performance of NMT (Xiao *et al.*, 2019). They propose the lattice graph, which incorporates different segmentation of source sentences. Shaw *et al.* (2018) construct the *relative position graph* to explicitly model the relative position feature. Yin *et al.* (2020) build the multi-modal graph to introduce visual knowledge to NMT, which presents the input sentences and corresponding images in a unified graph to capture the semantic correlations better. Despite the single-type static graph, Xu

et al. (2020c) construct the hybrid graph considering multiple relations for document-level NMT to address the severe long-dependency issue. In detail, they construct the graph considering both intra-sentential and inter-sentential relations. For intra-sentential relations, they link the words with sequential and dependency relations. For inter-sentential relations, they link the words in different sentences with lexical (repeated or similar) and coreference correlations.

- **Graph Representation Learning**. Most of the constructed graphs in this line are heterogeneous graphs, which contain multiple node or edge types and can't be exploited directly by typical GNNs. Thus, researchers adopt various heterogeneous graph representation techniques. Bastings *et al.* (2017) and Marcheggiani *et al.* (2018) regard the dependency graphs as multi-relational graphs and apply directed-GCN to learn the graph representation. Similarly, Beck *et al.* (2018b) firstly convert the constructed multi-relational graph to levi-graph and apply relational GGNN, which employs edge-type-specific parameters to exploit the rich structure information. Xu *et al.* (2020c) regard the edge as connectivity and treat the edge direction as edge types, such as "in", "out", and "self". Then they apply relational GCN to encode the document graph. Guo *et al.* (2019c) convert the heterogeneous graph to levi-graph and adopt the densely connected GCN to learn the embedding. Song *et al.* (2019) propose a special type-aware heterogeneous GGNN to learn the node embedding and edge representation jointly. Specifically, they first learn the edge representation by fusing both the source node and edge type's embeddings. Then for each node, they aggregate the representation from its incoming and outgoing neighbors and utilize a RNN based module to update the representation.

  Besides the extension of traditional GNNs, Transformer is further explored to learn from the structural inputs in NMT. Unlike traditional Transformer which adopt absolute sinusoidal position embedding to ensure the self-attention learn the position-specific feature, Shaw *et al.* (2018) and Xiao *et al.* (2019) adopt position-based edge embedding to capture the position correlations and make the transformer learn from the graph-based inputs. Cai and Lam (2020b) learn the bidirectional path-based relation embedding and add it to the node embedding when

calculating self-attention. They then find the shortest path from the given graph for any two nodes and apply bidirectional GRU to further encode the path to get the relation representation. Yin *et al.* (2020) apply graph-transformer-based encoder to learn the multi-modal graph. Firstly, for text modal's nodes, they get the initial embedding by summing up the word embedding and position embedding. As for visual nodes, they apply a MLP layer to project them to the unified space as text nodes. Secondly for each modal, they apply multi-head self-attention to learn the intra-modal representation. Thirdly, they employ GAT-based cross-modal fusion to learn the cross-modal representation.

- **Special Techniques**. In order to allow information flow from both directions, some technqies are designed for incorporating direction information. For example, Bastings *et al.* (2017), Marcheggiani *et al.* (2018), and Beck *et al.* (2018b) add the corresponding reverse edge as an additional edge type "reverse". The self-loops edge type are also added as type "self". For another example, Guo *et al.* (2019c) first add a global node and the edges from this global node to other nodes are marked with type "global". In addition, they further add bidirectional sequential links with type "forward" and "backward" between nodes existing in the input texts.

Benchmarks and Evaluation Common benchmarks for NMT from text include News Commentary v11, WMT14, WMT16, WMT19 for training, newstest2013, newstest2015, newstest2016, newsdev2019, newstest2019 for evaluation and testing. As for multi-modal NMT task, Multi30K dataset (Elliott *et al.*, 2016) is widely used by previous works. As for evaluation metrics, BLEU is the a typical metric to evaluate the similarity between the generated and real output texts.

Summarization

Background and Motivation Automatic summarization is the task of producing a concise and fluent summary while preserving key information content and overall meaning (Allahyari *et al.*, 2017). It is a well-noticed but challenging problem due to the need of searching in overwhelmed textural

data in real world. Broadly, there are two main classic settings in this task: 1) extractive summarization and 2) abstractive summarization. Extractive summarization task focus on selecting sub-sentences from the given text to reduce redundancy, which is formulated as a classification problem. In contrast, abstractive summarization follows the neural language generation task. It normally adopts the encoder-decoder architecture to generate the textual summary. Compared to the extractive summarization, the abstractive summarization setting is more challenging but more attractive since it can produce non-existing expressions. Traditional approaches (Gehrmann *et al.*, 2018; Zhou *et al.*, 2018b; Liu, 2019) simply regard the inputs as sequences and apply the encoder like LSTM, Transformer, etc. to learn the latent representation, which fail to utilize the rich structural information implicitly existing in the natural inputs. Many researchers find that structural knowledge is beneficial to address some troublesome challenges, e.g., long-dependency problem, and thus propose the GNN-based techniques (Wang *et al.*, 2020a; Fernandes *et al.*, 2019) to explicitly leverage the structural information to boost the performance.

Methodologies Most GNN-based summarization approaches firstly construct the graph to represent the given natural texts. Then they employ GNN-based encoders to learn the graph representation. After that, for extractive summarization models, they adopt the classifier to select candidate subsentences to compose the final summary. As for abstractive summarization, they mostly adopt the language decoder with maximizing the outputs' likelihood to generate the summary. In the following, we introduce some representative GNN-related techniques from the recent summarization methods.

- **Graph Construction**. Here we introduce the different ways to construct suitable and effective graph for different types of inputs, including sign-documents, multi-documents and codes.

  Single-document based. Fernandes *et al.* (2019) construct the hybrid graph, including sequential and coreference relation. To tackle the issue such as semantic irrelevance and deviation, Jin *et al.* (2020b) construct the semantic dependency graph and cast it as the multi-relational graph for the given texts. To capture the typical long-dependency in document-level summarization, Xu *et al.* (2020a) construct the hybrid graph. They

first construct the discourse graph by RST parsing and then add co-reference edges between co-reference mentions in the document. To better capture the long-dependency relation in sentence-level and enrich the semantic correlations, Wang *et al.* (2020a) regards both the sentences and the containing words as nodes and construct a similarity graph to model the semantic relations. To model the redundant relation between sentences, Jia *et al.* (2020) propose to construct the hybrid heterogeneous graph containing three types of nodes: 1) named entity, 2) word, and 3) sentence as well as four types of edges: 1) sequential, 2) containing, 3) same, and 4) similar. However, the methods above are mostly focused on the cross-sentence relations and overlook the inter-sentence, especially the topic information. To this end, Cui *et al.* (2020b) and Zhao *et al.* (2020a) construct the topic graph by introducing additional topic words to discover the latent topic information. On top of that, Zhao *et al.* (2020a) mine the sub-graph of non-topic nodes to represent the original texts while preserving the topic information.

Multi-document based. Yasunaga *et al.* (2017) decompose the given document clusters into sentences and construct the discourse graph by Personalized Discourse Graph algorithm (PDG). Li *et al.* (2020b) split the documents into paragraphs and constructs three individual graphs: 1) similarity graph, 2) discourse graph, and 3) topic graph to investigate the effectiveness.

Code based. To fully represent the code information in the code summarization task, Fernandes *et al.* (2019) construct the specific code graph for the given program clips. They first break up the identifier tokens (i.e., variables, methods, etc.) into sub-tokens by programming language heuristics. Then they construct the graph to organize the sub-tokens according to sequential positions and lexically usage. LeClair *et al.* (2020) propose another way by firstly parsing the given programs into abstract syntax trees (AST) and then converting them to program graphs.

- **Graph Representation Learning**

  In the literature of summarization tasks, both homogeneous GNNs and heterogeneous GNNs have been explored to learn the graph representation. For homogeneous graphs, Li *et al.* (2020b) apply self-attention-

based GAT to learn the representation on the fully-connected graph. Specifically, they introduce Gaussian kernel to mine the edge importance between nodes from the graph's topology. Zhao *et al.* (2020a) adopt the GAT-based graph transformer, which regards the similarity learned by self-attention as edge weight. For heterogeneous graphs, some researchers cast the heterogeneous graphs to homogeneous graphs by special techniques. For example, some works(LeClair *et al.*, 2020; Yasunaga *et al.*, 2017; Xu *et al.*, 2020a) ignore both the edges and nodes' types by treating the edge as connectivity. Cui *et al.* (2020b) project the nodes to the unified embedding space to diminish the heterogeneity. After that, some classic GNNs are employed such as GCN (Xu *et al.*, 2020a; LeClair *et al.*, 2020; Yasunaga *et al.*, 2017), GAT (Cui *et al.*, 2020b). For example, Fernandes *et al.* (2019) employ the relational GGNN to learn type-specific relations between nodes. Wang *et al.* (2020a) and Jia *et al.* (2020) firstly split the heterogeneous graph into two sub-graphs according to nodes' type (i.e., words graph and sentence graph) and then apply GAT-based cross-attention on two sub-graphs to learn the representation iteratively.

- **Embedding Initialization** The quality of the initial node embedding plays an important role in the overall performance of GNN-based methods. For graphs whose nodes are words, most approaches adopt the pre-trained word embeddings such as BERT (Li *et al.*, 2020b; Xu *et al.*, 2020a; Cui *et al.*, 2020b), ALBERT (Jia *et al.*, 2020). Besides, since the topic graph (Cui *et al.*, 2020b) introduces additional topic nodes, they initialize them by the latent representation of topic modeling. Jin *et al.* (2020b) apply Transformer to learn the contextual-level node embedding. For nodes such as sentence-level nodes, which are composed of words, Yasunaga *et al.* (2017) adopt the GRU to learn the sentences' embedding (i.e., the node embeddings) from the corresponding word sequences. They adopt the last hidden state as the sentences' representation. Similarly, Wang *et al.* (2020a) adopt CNN to capture the fine-grained n-gram feature and then employ Bi-LSTM to get the sentences' feature vectors. Jia *et al.* (2020) apply the average pooling function to the ALBERT's encoder outputs to represent the sentence nodes, while Zhao *et al.* (2020a) initialize the nodes (utterances) by

CNN and the topic words by LDA.

Benchmarks and Evaluation Common benchmarks for automatic summarization from documents include CNN/DailyMail (See *et al.*, 2017), NYT (Sandhaus, 2008), WikiSum (Liu *et al.*, 2018a), MultiNews (Fabbri *et al.*, 2019). As for code based summarization, Java (Alon *et al.*, 2018) and Python (Barone and Sennrich, 2017) are widely used. As for evaluation metrics, BLEU, ROUGE and human evaluation are commonly used.

Structural-data to Text

Background and Motivation Despite the natural texts, many NLP applications evolve the data which is represented by explicit graph structure, such as SQL queries, knowledge graphs, AMR, etc. The task of structural-data is to generate the natural language from structural-data input. Traditional works (Pourdamghani *et al.*, 2016; Pourdamghani *et al.*, 2014) apply the linearization mechanisms which map the structural-data to sequential data and adopt the Seq2Seq architecture to generate the texts. To fully capture the rich structure information, recent efforts focus on GNN-based techniques to handle this task. In the following, we introduce GNN techniques for three typical cases, namely AMR-to-text generation, SQL-to-text generation and RDF-to-text generation.

Methodologies Most GNN-based AMR-to-text and SQL-to-text approaches typically construct domain-specific graphs such as AMR graphs and SQL-parsing-based graphs to organize the inputs. RDF-to-text generation often uses the graph structure inherent in the RDF triples. Following that, they apply Graph2Seq consisting of GNN encoders and sequential decoders to generate neural language outputs. This section summarizes various graph construction methods and the techniques employed to exploit the informative graphs.

- **Graph Construction**. Regarding the AMR-to-text Generation, the input AMRs can be normally represented as directed heterogeneous graphs according to the relations (Damonte and Cohen, 2019; Song *et al.*, 2020; Bai *et al.*, 2020; Zhu *et al.*, 2019b; Zhang *et al.*, 2020d; Yao *et al.*, 2020; Beck *et al.*, 2018b; Cai and Lam, 2020b; Jin and Gildea, 2020; Ribeiro

et al., 2019a; Wang *et al.*, 2020g; Wang *et al.*, 2020f; Song *et al.*, 2018c). To incorporate the conventional GNNs specializing in homogeneous-graph learning, Damonte and Cohen (2019), Yao *et al.* (2020), Beck *et al.* (2018b), Cai and Lam (2020a), and Ribeiro *et al.* (2019a) convert the AMR graphs to levi-graph. In addition, for each edge, they (Damonte and Cohen, 2019; Beck *et al.*, 2018b; Yao *et al.*, 2020; Cai and Lam, 2020a) add the reverse edges and self-loops to allow information flows in both directions. Besides the default, reverse, and self-loop edges, Yao *et al.* (2020) also introduces fully-connected edges to model indirect nodes and connected edges, which treat original edges as connectivity without direction to model connection information. Zhao *et al.* (2020b) split the given AMR graph $\mathcal{G}_{AMR}$ into two directed sub-graphs: 1) concept graph $\mathcal{G}_c$, and 2) line graph $\mathcal{G}_l$. They firstly treat the edge as connectivity to get the concept graph. Then for each edge in $\mathcal{G}_{AMR}$, they create a node in $\mathcal{G}_l$. Two nodes in $\mathcal{G}_l$ are connected if they share the same nodes in $\mathcal{G}_{AMR}$. The two sub-graphs are connected by original connections in $\mathcal{G}_{AMR}$. To leverage multi-hop connection information, they preserve the $1 - K$ order neighbors in the adjacency matrices. Regarding the SQL inputs, the SQL queries can be parsed by many SQL tools[1] into many sub-clauses without loss, which naturally contain rich structure information. Xu *et al.* (2018a) and Xu *et al.* (2018b) construct the directed and homogeneous SQL-graph based on the sub-clauses by some hand-craft rules. Regarding the RDF triple inputs, Marcheggiani and Perez-Beltrachini (2018) and Gao *et al.* (2020) treat the relation in a triple as an additional node in the graph connecting to the subject and object entity nodes.

- **Graph Representation Learning**. Ribeiro *et al.* (2019a) and Gao *et al.* (2020) treat the obtained levi-graphs as directed homogeneous graphs and learn the representation by bidirectional GNNs. Ribeiro *et al.* (2019a) also proposes a bidirectional embedding learning framework that traverses the directed graphs in the original and the reversal direction. Xu *et al.* (2018a) apply classic graph2seq architecture (Xu *et al.*, 2018b) with bidirectional GraphSage methods to learn the embedding of SQL graph via two ways, including 1) pooling-based mechanism

[1] http://www.sqlparser.com.

and 2) node-based mechanism, which means add a supernode connecting to other nodes, to investigate the influence of graph embedding. Some approaches directly employ multi-relational GNN to encode the obtained multi-relational graphs. For example, Damonte and Cohen (2019) adopt directed-GCN to exploit the AMR graphs considering both heterogeneity and parameter-overhead. Beck *et al.* (2018b) propose relational GGNN to capture diverse semantic correlations. Song *et al.* (2018c) employ a variance of GGNN to exploit the multi-relational AMR graphs by aggregating the bidirectional node and edge features and then fusing them via a LSTM network. Zhao *et al.* (2020b) propose a heterogeneous GAT to exploit the AMR graphs in different grains. Firstly, they apply GAT to each sub-graph to learn the bidirectional representation separately. Then they apply cross-attention to explore the dependencies between the two sub-graphs. Zhang *et al.* (2020d) propose the multi-hop GCN, which dynamically fuses the $1 - K$ order neighbors' features to control the information propagate in a range of orders. Wang *et al.* (2020g) apply relational GAT with bidirectional graph embedding mechanism by incorporating the edge types into the attention procedure to learn type-specific attention weights.

Transformer architectures are also utilized to encode the AMR or SQL graphs. Yao *et al.* (2018) firstly apply GAT-based graph Transformer in each homogeneous sub-graphs and then concatenate sub-graphs representation to feed the feed-forward layer. Some works Zhu *et al.* (2019b), Song *et al.* (2020), Bai *et al.* (2020), Cai and Lam (2020a), and Jin and Gildea (2020) adopt the structure-aware graph transformer (Zhu *et al.*, 2019b; Cai and Lam, 2020b), which injecting the relation embedding learned by shortest path to the self-attention to involve the structure features. Specifically, Jin and Gildea (2020) explore various shortest path algorithms to learn the relation representation of arbitrary two nodes. Similarly, Wang *et al.* (2020f) employ the graph Transformer, which leverages the structure information by incorporating the edge types into attention-weight learning formulas.

- **Special Mechanisms**. Damonte and Cohen (2019) apply the Bi-LSTM encoder following the GNN encoder to further encode the sequential information. Despite the language generation procedure, to better pre-

serve the structural information, Zhu *et al.* (2019b), Bai *et al.* (2020), and Wang *et al.* (2020g) introduce the graph reconstruction on top of the latent graph representation generated by graph transformer encoder.

Benchmarks and Evaluation Common benchmarks for AMR-to-text generation task include LDC2015E85, LDC2015E86, LDC2017T10, and LDC2020T02. As for the SQL-to-text generation task, WikiSQL (Zhong *et al.*, 2017) and Stackoverflow (Iyer *et al.*, 2016) are widely used by previous works. The RDF-to-text generation task often uses WebNLG (Gardent *et al.*, 2017) and New York Times (NYT) (Riedel *et al.*, 2010). As for evaluation metrics, the AMR-to-text generation task mostly adopts BLEU, Meteor, CHRF++, and human evaluation including meaning similarity and readability. While BLEU-4 are widely used for SQL-to-text task. The RDF-to-text generation task uses BLEU, Meteor and TER.

Natural Question Generation

Background and Motivation The natural question generation (QG) task aims at generating natural language questions from certain form of data, such as KG (Kumar *et al.*, 2019; Chen *et al.*, 2020g), tables (Bao *et al.*, 2018), text (Du *et al.*, 2017; Song *et al.*, 2018a) or images (Li *et al.*, 2018b), where the generated questions need to be answerable from the input data. Most prior work (Du *et al.*, 2017; Song *et al.*, 2018a; Kumar *et al.*, 2019) adopts a Seq2Seq architecture which regards the input data as sequential data without considering its rich structural information. For instance, when encoding the input text, most previous approaches (Du *et al.*, 2017; Song *et al.*, 2018a) typically ignore the hidden structural information associated with a word sequence such as the dependency parsing tree. Even for the setting of QG from KG, most approaches (Kumar *et al.*, 2019) typically linearize the KB subgraph to a sequence and apply a sequence encoder. Failing to utilize the graph structure of the input data may limit the effectiveness of QG models. As for the multi-hop QG from text setting which requires reasoning over multiple paragraphs or documents, it is beneficial to capture the relationships among different entity mentions across multiple paragraphs or documents. In summary, modeling the rich structures of the input data is important for many QG tasks. Recently, GNNs have been successfully applied to the QG

tasks (Liu *et al.*, 2019b; Chen *et al.*, 2020i; Wang *et al.*, 2020d).

Methodologies Most GNN-based QG approaches adopt a Graph2Seq architecture where a GNN-based encoder is employed to model the graph-structured input data, and a sequence decoder is employed to generate a natural language question. In this section, we introduce and summarize some representative GNN-related techniques adopted in recent QG approaches.

- **Graph Construction**. Different graph construction strategies have been proposed to suit the various needs of different QG settings by prior GNN-based approaches. Some works (Liu *et al.*, 2019b; Wang *et al.*, 2020d; Chen *et al.*, 2020i; Pan *et al.*, 2020) converted the passage text to a graph based on dependency parsing or semantic role labeling for QG from text. As for multi-hop QG from text, in order to model the relationships among entity mentions across multiple paragraphs or documents, an entity graph is often constructed. For instance, Su *et al.* (2020) constructed an entity graph with the named entities in context as nodes and edges connecting the entity pairs appearing in the same sentence or paragraph. In addition, an answer-aware dynamic entity graph was created on the fly by masking out entities irrelevant to the answers. Sachan *et al.* (2020) built a so-called context-entity graph containing three types of nodes (i.e., named-entity mentions, coreferent entities, and sentence-ids) and added edges connecting them. Unlike the above approaches that build a static graph based on prior knowledge, Chen *et al.* (2020i) explored dynamic graph construction for converting the passage text to a graph of word nodes by leveraging the attention mechanism. As for QG from KG, graph construction is not needed since the KG is already provided. A common option is to extract a k-hop subgraph surrounding the topic entity as the input graph when generating a question (Chen *et al.*, 2020g).

- **Graph Representation Learning**. Common GNN models used by existing QG approaches include GCN (Liu *et al.*, 2019b; Su *et al.*, 2020), GAT (Wang *et al.*, 2020d), GGNN (Chen *et al.*, 2020i; Chen *et al.*, 2020g; Pan *et al.*, 2020), and graph transformer (Sachan *et al.*, 2020). In order to model the edge direction information, Chen *et al.* (2020i)

and Chen *et al.* (2020g) extended the GGNN model to handle directed edges. In order to model multi-relational graphs, Chen *et al.* (2020g) explored two graph encoding strategies: i) converting a multi-relational graph to a Levi graph (Levi, 1942) and applying a regular GGNN model, or ii) extending the GGNN model by incorporating the edge information in the message passing process. Pan *et al.* (2020) also extended the GGNN model by bringing in the attention mechanism from GAT and introducing edge type aware linear transformations for message passing between node pairs. Sachan *et al.* (2020) proposed a graph-augmented transformer model employing a relation-aware multi-head attention mechanism similar to Zhu *et al.* (2019c) and Cai and Lam (2020b). Pan *et al.* (2020) and Sachan *et al.* (2020) found it beneficial to additionally model the sequential information in the input text besides the graph-structured information. Pan *et al.* (2020) separately applied a sequence encoder to the document text, and a graph encoder to the semantic graph representation of the document constructed from semantic role labeling or dependency parsing. The outputs of the sequence encoder and graph encoder would then be fused and fed to a sequence decoder for question generation. The model was jointly trained on question decoding and content selection sub-tasks. Sachan *et al.* (2020) ran both the structure-aware attention network on the input graph and the standard attention network on the input sequence, and fused their output embeddings using some non-linear mapping function to learn the final embeddings for the sequence decoder. During the training, a contrastive objective was proposed to predict supporting facts, serving as a regularization term in addition to the main cross-entropy loss for sequence generation.

Benchmarks and Evaluation Common benchmarks for QG from text include SQuAD (Rajpurkar *et al.*, 2016), NewsQA (Trischler *et al.*, 2017), and HotpotQA (Yang *et al.*, 2018a). As for QG from KG, WebQuestions (Kumar *et al.*, 2019) and PathQuestions (Kumar *et al.*, 2019) are widely used by previous works. As for evaluation metrics, BLEU-4, METEOR, ROUGE-L and human evaluation (e.g., syntactically correct, semantically correct, relevant) are common metrics. Complexity is also used to evaluate the performance of multi-hop QG systems.

0.7.2 Machine Reading Comprehension and Question Answering

Machine Reading Comprehension

Background and Motivation The task of Machine Reading Comprehension (MRC) aims to answer a natural language question using the given passage. Significant progress has been made in the MRC task thanks to the development of various (co-)attention mechanisms that capture the interaction between the question and context (Hermann *et al.*, 2015; Cui *et al.*, 2017; Seo *et al.*, 2017; Xiong *et al.*, 2017a). Considering that the traditional MRC setting mainly focuses on one-hop reasoning which is relatively simple, recently, more research efforts have been made to solve more challenging MRC settings. For instance, the multi-hop MRC task is to answer a natural language question using multiple passages or documents, which requires the multi-hop reasoning capacity. The conversational MRC task is to answer the current natural language question in a conversation given a passage and the previous questions and answers, which requires the capacity of modeling conversation history. The numerical MRC task requires the capacity of performing numerical reasoning over the passage. These challenging MRC tasks call for the learning capacity of modeling complex relations among objects. For example, it is beneficial to model relations among multiple documents and the entity mentions within the documents for the multi-hop MRC task. Recently, GNNs have been successfully applied to various types of MRC tasks including multi-hop MRC (Song *et al.*, 2018b; Cao *et al.*, 2019a; Qiu *et al.*, 2019; Cao *et al.*, 2019c; Fang *et al.*, 2020b; Tang *et al.*, 2020b; Zheng and Kordjamshidi, 2020; Tu *et al.*, 2019b; Ding *et al.*, 2019a), conversational MRC (Chen *et al.*, 2020h), and numerical MRC (Ran *et al.*, 2019).

Methodologies GNN-based MRC approaches typically operate by first constructing an entity graph or hierarchical graph capturing rich relations among nodes in the graph, and then applying a GNN-based reasoning module for performing complex reasoning over the graph. Assuming the GNN outputs already encode the semantic meanings of the node itself and its neighboring structure, a prediction module will finally be applied for predicting answers. The graph construction techniques and graph representation techniques developed for solving the MRC task vary between different approaches. In this section, we introduce and summarize some representative GNN-related

techniques adopted in recent MRC approaches.

- **Graph Construction**. In order to apply GNNs for complex reasoning in the MRC task, one critical step is graph construction. Building a high-quality graph capturing rich relations among useful objects (e.g., entity mentions, paragraphs) is the foundation for conducting graph-based complex reasoning. Most GNN-based MRC approaches conduct static graph construction by utilizing domain-specific prior knowledge. Among all existing GNN-based MRC approaches, the most widely adopted strategy for static graph construction is to construct an entity graph using carefully designed rules. These approaches (Song *et al.*, 2018b; Cao *et al.*, 2019a; Qiu *et al.*, 2019; Cao *et al.*, 2019c; Tang *et al.*, 2020b; Zheng and Kordjamshidi, 2020; Ran *et al.*, 2019) usually extract entity mentions from questions, paragraphs and candidate answers (if given) as nodes, and connect the nodes with edges capturing different types of relations such as exact match, co-occurrence, coreference and semantic role labeling. Edge connectivity with different granularity levels in terms of context window (e.g., sentence, paragraph and document) might also be distinguished for better modeling performance (Qiu *et al.*, 2019; Cao *et al.*, 2019c). For instance, Cao *et al.* (2019c) distinguished cross-document edge and within-document edge when building an entity graph. As for the numerical MRC task, the most important relations are probably the arithmetic relations. In order to explicitly model numerical reasoning, Ran *et al.* (2019) constructed a graph containing numbers in the question and passage as nodes, and added edges to capture various arithmetic relations among the numbers. Besides building an entity graph capturing various types of relations among entity mentions, some approaches (Tu *et al.*, 2019b; Fang *et al.*, 2020b; Zheng *et al.*, 2020) opt to build a hierarchical graph containing various types of nodes including entity mentions, sentences, paragraphs and documents, and connect these nodes using predefined rules. For example, Zheng *et al.* (2020) constructed a hierarchical graph that contains edges connecting token nodes and sentence nodes, sentence nodes and paragraph nodes as well as paragraph nodes and document nodes.

  Very recently, dynamic graph construction techniques without relying on hand-crafted rules have also been explored for the MRC task and

achieved promising results. Unlike static graph construction techniques that have been widely explored in the MRC literature, dynamic graph construction techniques are less studied. In comparison to static graph construction, dynamic graph construction aims to build a graph on the fly without relying on domain-specific prior knowledge, and is typically jointly learned with the remaining learning modules of the system. Recently, Chen *et al.* (2020h) proposed a GNN-based model for the conversational MRC task, which is able to dynamically build a question and conversation history aware passage graph containing each passage word as a node at each conversation turn by leveraging the attention mechanism. A kNN-style graph sparsification operation was conducted so as to further extract a sparse graph from the fully-connected graph learned by the attention mechanism. The learned sparse graph will be consumed by the subsequent GNN-based reasoning module, and the whole system is end-to-end trainable.

- **Graph Representation Learning**. Most GNN-based MRC approaches rely on a GNN model for performing complex reasoning over the graph. In the literature of the MRC task, both homogeneous GNNs and multi-relational GNNs have been explored for node representation learning. Even though most GNN-based MRC approaches construct a multi-relational or heterogeneous graph, some of them still apply a homogeneous GNN model such as GCN (Zheng and Kordjamshidi, 2020; Ding *et al.*, 2019a), GAT (Qiu *et al.*, 2019; Fang *et al.*, 2020b; Zheng *et al.*, 2020) and Graph Recurrent Network (GRN) (Song *et al.*, 2018b). Unlike other works that apply a GNN model to a single graph, Chen *et al.* (2020h) proposed a Recurrent Graph Neural Network (RGNN) for processing a sequence of passage graphs for modeling conversational history. The most widely used multi-relational GNN model in the MRC task is the RGCN model (Schlichtkrull *et al.*, 2018). Many approaches (Cao *et al.*, 2019a; Cao *et al.*, 2019c; Tu *et al.*, 2019b; Ran *et al.*, 2019) adopt a gating RGCN variant which in addition introduces a gating mechanism regulating how much of the update message propagates to the next step. Tang *et al.* (2020b) further proposed a question-aware gating mechanism for RGCN, that is able to regulate the aggregated message according to the question, and even bring the

question information into the update message.

- **Node Embedding Initialization**. Many studies have shown that the quality of the initial node embeddings play an important role in the overall performance of GNN-based models. Most approaches use pre-trained word embeddings such as GloVe (Pennington *et al.*, 2014), ELMo (Peters *et al.*, 2018), BERT (Devlin *et al.*, 2019) and RoBERTa (Liu *et al.*, 2019d) to initialize tokens. Some works (Cao *et al.*, 2019c; Chen *et al.*, 2020h) also concatenated linguistic features to word embeddings to enrich the semantic meanings. On top of the initial word embeddings, most approaches choose to further apply some transformation functions such as MLP for introducing nonlinearity (Tang *et al.*, 2020b), BiLSTM for capturing local dependency of the text (Cao *et al.*, 2019a; Chen *et al.*, 2020h; Fang *et al.*, 2020b), co-attention layer for fusing questions to passages (Qiu *et al.*, 2019; Tu *et al.*, 2019b; Chen *et al.*, 2020h; Fang *et al.*, 2020b).

- **Special Techniques** In order to increase the richness of the supervision signals, some approaches adopt the multi-tasking learning strategy to predict not only the answer span, but also the supporting paragraph/sentence/fact and answer type (Qiu *et al.*, 2019; Fang *et al.*, 2020b; Zheng and Kordjamshidi, 2020; Chen *et al.*, 2020h).

Benchmarks and Evaluation Common multi-hop MRC benchmarks include HotpotQA (Yang *et al.*, 2018a), WikiHop (Welbl *et al.*, 2018) and ComplexWebQuestions (Talmor and Berant, 2018). Common conversational MRC benchmarks include CoQA (Reddy *et al.*, 2019), QuAC (Choi *et al.*, 2020) and DoQA (Campos *et al.*, 2019). DROP (Dua *et al.*, 2019) is a benchmark created for the numerical MRC task. As for evaluation metrics, F1 and EM (i.e., exact match) are the two most widely used evaluation metrics for the MRC task. Besides, the Human Equivalence Score (HEQ) (Choi *et al.*, 2020; Campos *et al.*, 2019) is used to judge whether a system performs as well as an average human. HEQ-Q and HEQ-D are accuracies at the question level and dialog level, respectively.

Knowledge Base Question Answering

Background and Motivation Knowledge Base Question Answering (KBQA) has emerged as an important research topic in the past few years (Yih *et al.*, 2015; Zhang *et al.*, 2018d; Chen *et al.*, 2019). The goal of KBQA is to automatically find answers from the KG given a natural language question. Recently, due to its nature capability of modeling relationships among objects, GNNs have been successfully applied for performing the multi-hop KBQA task which requires reasoning over multiple edges of the KG to arrive at the right answer. A relevant task is open domain QA (Sun *et al.*, 2018a; Sun *et al.*, 2019b) which aims to answer open domain questions by leveraging hybrid knowledge sources including corpus and KG. Here we only focus on the QA over KG setting, while the GNN applications in the open domain QA task will be introduced in other sections.

Methodologies In this section, we introduce and summarize some representative GNN-related techniques adopted in the recent KBQA research.

- **Graph Construction**. Semantic parsing based KBQA methods (Yih *et al.*, 2015) aims at converting natural language questions to a semantic graph which can be further executed against the KG to find the correct answers. In order to better model the structure of the semantic graph, Sorokin and Gurevych (2018a) proposed to use GNNs to encode the candidate semantic graphs. Specifically, they used a similar procedure as Yih *et al.* (2015) to construct multiple candidate semantic graphs given the question, and chose the one which has the highest matching score to the question in the embedding space. Feng *et al.* (2020b) and Yasunaga *et al.* (2021) focused on a different multi-choice QA setting which is to select the correct answer from the provided candidate answer set given the question and the external KG. For each candidate answer, Feng *et al.* (2020b) proposed to extract from the external KG a "contextualized" subgraph according to the question and candidate answer. This subgraph serves as the evidence for selecting the corresponding candidate answer as the final answer. Specifically, they first recognized all the entity mentions in the question and candidate answer set, and linked them to entities in the KG. Besides these linked entities in the KG, any other

entities that appeared in any two-hop paths between pairs of mentioned entities in the KG as well as the corresponding edges were also added to the subgraph. Yasunaga *et al.* (2021) constructed a joint graph by regarding the QA context as an additional node (QA context node) and connecting it to the topic entities in the KG subgraph. Specifically, they introduced two new relation types $r_{z,q}$ and $r_{z,a}$ for capturing the relationship between the QA context node and the relevant entities in the KG. The specific relation type is determined by whether the KG entity is found in the question portion or the answer portion of the QA context.

- **Graph Representation Learning** In order to better model the constructed multi-relational or heterogeneous graphs, basic GNNs need to be extended to handle edge types or node types. To this end, Sorokin and Gurevych (2018a) extended GGNN (Zhang *et al.*, 2020e) to include edge embeddings in message passing. After learning the vector representations of both the question and every candidate semantic graph, they used a simple reward function to select the best semantic graph for the question. The final node embedding of the question variable node (q-node) in each semantic graph was extracted and non-linearly transformed to obtain the graph-level representation. Feng *et al.* (2020b) designed a Multi-hop Graph Relation Network (MHGRN) to unify both GNNs and path-based models. Specifically, they considered both node type and edge type information of the graph by introducing node type specific linear transformation, and node type and relation type aware attention in message passing. In addition, instead of performing one hop message passing at each time, inspired by path-based models (Santoro *et al.*, 2017; Lin *et al.*, 2019b), they proposed to pass messages directly over all the paths of lengths up to K. Graph-level representations were obtained via attentive pooling over the output node embeddings, and would be concatenated with the text representation of question and each candidate answer to compute the plausibility score. Similarly, Yasunaga *et al.* (2021) extended GAT by introducing node type and edge type aware message passing to handle multi-relational graphs. They in addition employed a pre-trained language model for KG node relevance scoring in the initial stage and final answer selection stage.

Benchmarks and Evaluation Common benchmarks for KBQA include WebQuestionsSP (Yih *et al.*, 2016), MetaQA (Zhang *et al.*, 2018d), QALD-7 (Usbeck *et al.*, 2017), CommonsenseQA (Talmor *et al.*, 2019), and OpenbookQA (Mihaylov *et al.*, 2018). F1 and accuracy are common metrics for evaluating KBQA methods.

Open-domain Question Answering

Background and Motivation The task of open-domain question answering aims to identify answers to the natural question given a large scale of open-domain knowledge (e.g. documents, knowledge base and etc.). Untill recent times, the open-domain question answering (Bordes *et al.*, 2015; Zhang *et al.*, 2018d) has been mostly exploited through knowledge bases such as Personalized PageRank (Haveliwala, 2002), which actually closely related to the Knowledge based Question Answering task (KBQA) in techniques. The knowledge based methods benefit from obtaining external knowledge easily through graph structure. However, these methods limit in the missing information of the knowledge base and fixed schema. Other attempts have been made to answer questions from massive and unstructured documents (Chen *et al.*, 2017a). Compared to the KB based methods, these methods can fetch more information but suffer from the difficulty of retrieve relevant and key information from redundant external documents.

Methodologies In this section, we introduce and summarize some representative GNN-related techniques in the recent open-domain question answering research.

- **Graph Construction**. Most of the GNN based methods address the mentioned challenges by constructing a heterogeneous graph with both knowledge base and unstructured documents (Han *et al.*, 2020; Sun *et al.*, 2018a; Sun *et al.*, 2019b). Han *et al.* (2020) and Sun *et al.* (2018a) firstly extract the subgraph from external knowledge base named Personalized PageRank (Haveliwala, 2002). Then they fetch a relevant text corpus from Wikipedia and fuse them to the knowledge graph. Specifically, they represent the documents by words' encoding and link the nodes (the nodes in the knowledge graph are entities) which appear

in the document. Sun *et al.* (2019b) propose a iteratively constructed heterogeneous graph method from both knowledge base and text corpus. Initially, the graph depends only on the question. Then for each iteration, they expand the subgraph by choosing nodes from which to "pull" information about, from the KB or corpus as appropriate.

- **Graph Representation Learning** Sun *et al.* (2018a) and Sun *et al.* (2019b) first initialize the nodes' embedding with pre-trained weight for entities and LSTM encoding for documents. They further propose different update rule for both entities and documents. For entities, they apply R-GCN (Schlichtkrull *et al.*, 2018) on the sub-graph only from the knowledge base and then take average of the linked words' feature in the connected documents. The entities' representation is the combination of: 1) the previous entities themselves' representation, 2) question encoding, 3) knowledge-subgraph's aggregation results, and 4) related documents' aggregation results. For documents' update operation, they aggregate the features from connected entities. Sun *et al.* (2018a) adopt similar idea for heterogeneous graph representation learning. Technically, before encoding entities, they incorporate the connected words' embedding in the documents to the entities. Then for nodes, they propose GCN with attention weight to aggregate neighbor entities. Note that the question is employed in the attention mechanism to guide the learning process. The documents' updating process is in the same pattern.

Benchmarks and Evaluation Common benchmarks for Open-domain Question answering include WebQuestionsSP (Yih *et al.*, 2016), MetaQA (Zhang *et al.*, 2018d), Complex WebQuestions 1.1 (Complex WebQ) (Talmor and Berant, 2018), and WikiMovies-10K (Miller *et al.*, 2016). Hits@1 and F1 scores are the common evaluation metrics for this task (Sun *et al.*, 2018a; Sun *et al.*, 2019b; Han *et al.*, 2020).

Community Question Answering

Background and Motivation The task of community question answering aims to retrieve the relevant answer from QA forums such as Stack Overflow or Quora. Different from the traditional MRC (QA) task, CQA systems are able

to harness tacit knowledge (embedded in their diverse communities) or explicit knowledge (embedded in all resolved questions) in answering of an enormous number of new questions posted each day. Nevertheless, the growing number of new questions could make CQA systems without appropriate collaboration support become overloaded by users' requests.

Methodologies In this section, we introduce and summarize some representative GNN-related techniques adopted in the recent CQA research.

- **Graph Construction**. Most of the GNN-based methods construct a multi-modal graph for existing question/answer pairs (Hu *et al.*, 2019a; Hu *et al.*, 2020a). For the given q/a pair (q, a), both of them construct the question/answer $\mathcal{G}^q/\mathcal{G}^a$ graph separately. Since in real community based forums, the question/answer pairs may contain both visual and text contents, they employ a multi-modal graph to represent them jointly. Hu *et al.* (2019a) firstly employ object detection models such as YOLO3 (Redmon and Farhadi, 2018) to fetch visual objects. The objects are represented by their labels (visual words more accurately). The visual objects are treated as words in the answers which are modeled with textural contents equally. Then they regard each textural words as vertex and link them with undirected occurrence edges. Hu *et al.* (2020a) adopt the same idea as (Hu *et al.*, 2019a) for building occurrence graph for both textural contents and visual words. But for extracting visual words from images, they employ unsupervised Meta-path Link Prediction for Visual Labeling. Concretely, they define the meta-path over image and words and build the heterogeneous image-word graph.

- **Graph Representation Learning**. Most of the GNN-based community question answering models adapt the GNN models to capture structure information. Given the question/answer pair (q, a), Hu *et al.* (2019a) stacks the graph pooling network to capture the hierarchical semantic-level correlations between nodes. Conceptually, the graph pooling network extract the high-level semantic representation for both question and answer graphs. Formally, it consists of two GCN-variant APPNP (Klicpera *et al.*, 2019) encoders. Generally, one APPNP is employed to learn the high-level semantic cluster distribution for each vertex. The other APPNP network is used to learn the immediate node

representation. The final node representation is the fusion of the two encoders' results. Hu *et al.* (2020a) employ the APPNP to learn the importance of each vertex's neighbors.

Benchmarks and Evaluation Common benchmarks for Community Question Answering include Zhihu and Quora released by MMAICM (Hu *et al.*, 2018). The normalized discounted cumulative gain (nDCG) and precision are common metrics for evaluating Community Question Answering methods (Hu *et al.*, 2018; Hu *et al.*, 2020a; Hu *et al.*, 2019a).

0.7.3 Dialog Systems

Background and Motivation Dialog system (Williams *et al.*, 2014; Chen *et al.*, 2017b) is a computer system that can continuously converse with a human. In order to build a successful dialog system, it is important to model the dependencies among different interlocutors or utterances within a conversation. Due to the ability of modeling complex relations among objects, recently, GNNs have been successfully applied to various dialog system related tasks including dialog state tracking (Chen *et al.*, 2018b; Chen *et al.*, 2020b) which aims at estimating the current dialog state given the conversation history, dialog response generation (Hu *et al.*, 2019e) which aims at generating the dialog response given the conversation history, and next utterance selection (Liu *et al.*, 2021c) which aims at selecting the next utterance from a candidate list given the conversation history.

Methodologies In this section, we introduce and summarize some representative GNN-related techniques adopted in the recent dialog systems research.

- **Graph Construction**. Building a high-quality graph representing a structured conversation session is challenging. A real-world conversation can have rich interactions among speakers and utterances. Here, we introduce both static and dynamic graph construction techniques used in recent GNN-based approaches. For static graphs, most GNN-based dialog systems rely on prior domain knowledge to construct a graph. For instance, in order to apply GNNs to model multi-party dialogues (i.e.,

involving multiple interlocutors), Hu *et al.* (2019e) converted utterances in a structured dialogue session to a directed graph capturing response relationships between utterances. Specifically, they created an edge for every pair of utterances from the same speaker following the chronological order of the utterances. Chen *et al.* (2018b) built a directed heterogeneous graph according to the domain ontology that consists of edges among slot-dependent nodes and slot-independent nodes. Chen *et al.* (2020b) constructed three types of graphs including a token-level schema graph according to the original ontology scheme, a utterance graph according to the dialogue utterance, and a domain-specific slot-level schema graph connecting two slots from the same domain or share the same candidate values. Liu *et al.* (2021c) constructed a graph connecting utterance nodes that are adjacent or belong to dependent topics. Regarding the dynamic graph construction, unlike most GNN-based approaches that rely on prior knowledge for constructing static graph, Chen *et al.* (2018b) jointly optimized the graph structure and the parameters of GNN by approximating posterior probability of the adjacency matrix (i.e., modeled as a latent variable following a factored Bernoulli distribution) via variational inference (Hoffman *et al.*, 2013).

- **Graph Representation Learning** Various GNN models have been applied in dialog systems. For instance, Liu *et al.* (2021c) applied GCN to facilitate reasoning over all utterances. Chen *et al.* (2020b) proposed a graph attention matching network to learn the representations of ontology schema and dialogue utterance simultaneously, and a recurrent attention graph neural network which employs a GRU-like gated cell for dialog state updating. Inspired by the hierarchical sequence-based HRED model for dialog response generation, Hu *et al.* (2019e) proposed an utterance-level graph-structured encoder which is a gated GNN variant, and is able to control how much the new information (from the preceding utterance nodes) should be considered when updating the current state of the utterance node. They also designed a bi-directional information flow algorithm to allow both forward and backward message passing over the directed graph. In order to model multi-relational graphs, Chen *et al.* (2018b) designed a R-GCN like GNN model employing edge type specific weight matrices.

- **Node Embedding Initialization** In terms of node embedding initialization for GNN models, Hu *et al.* (2019e) applied a BiLSTM to first encode the local dependency information in the raw text sequence. Liu *et al.* (2021c) used the state-of-the-art pre-trained ALBERT embeddings (Lan *et al.*, 2019) to initialize the node embeddings. Chen *et al.* (2020b) included token embeddings, segmentation embeddings as well as position embeddings to capture the rich semantic meanings of the nodes.

Benchmarks and Evaluation Common dialog state tracking benchmarks include PyDial (Casanueva *et al.*, 2017) and MultiWOZ (Budzianowski *et al.*, 2018; Eric *et al.*, 2020). Ubuntu Dialogue Corpus (Lowe *et al.*, 2015) and MuTual (Cui *et al.*, 2020a) are often used for evaluating dialog response generation and next utterance selection systems. As for evaluation metrics, BLEU, METEOR and ROUGE-L are common metrics for evaluating dialog response generation systems. Besides automatic evaluation, human evaluation (e.g., grammaticality, fluency, rationality) is often conducted. Accuracy is the most widely used metric for evaluating dialog state tracking systems. Recall at k is often used in the next utterance selection task.

0.7.4 Text Classification

Background and Motivation Traditional text classification methods heavily rely on feature engineering (e.g., BOW, TF-IDF or more advanced graph path based features) for text representation. In order to learn "good" representations from text, various unsupervised approaches have been proposed for word or document representation learning, including word2vec (Mikolov *et al.*, 2013), GloVe (Pennington *et al.*, 2014), topic models (Blei *et al.*, 2003; Larochelle and Lauly, 2012), autoencoder (Miao *et al.*, 2016; Chen and Zaki, 2017), and doc2vec (Le and Mikolov, 2014; Kiros *et al.*, 2015). These pre-trained word or document embeddings can further be consumed by a MLP (Joulin *et al.*, 2017), CNN (Kim, 2014) or LSTM (Liu *et al.*, 2016; Zhang *et al.*, 2018b) module for training a supervised text classifier. In order to better capture the relations among words in text or documents in corpus, various graph-based approaches have been proposed for text classification. For instance, Peng *et al.* (2018) proposed to first construct a graph of words, and

then apply a CNN to the normalized subgraph. Tang *et al.* (2015) proposed a network embedding based approach for text representation learning in a semi-supervised manner by converting a partially labeled text corpora to a heterogeneous text network. Recently, given the strong expressive power, GNNs have been successfully applied to both semi-supervised (Yao *et al.*, 2019a; Liu *et al.*, 2020; Hu *et al.*, 2019c) and supervised (Defferrard *et al.*, 2016; Huang *et al.*, 2019; Zhang *et al.*, 2020e) text classification.

Methodologies GNN-based text classification approaches typically operate by first constructing a document graph or corpus graph capturing rich relations among nodes in the graph, and then applying a GNN to learn good document embeddings which will later be fed into a softmax layer for producing a probabilistic distribution over a class of labels. The graph construction techniques and graph representation techniques developed for solving the text classification task vary between different approaches. In this section, we introduce and summarize some representative GNN-related techniques adopted in recent text classification approaches.

- **Graph Construction**. Semi-supervised text classification leverages a small amount of labeled data with a large amount of unlabeled data during training. Utilizing the relations among labeled and unlabeled documents is essential for performing well in this semi-supervised setting. Regarding the static graph construction, recently, many GNN-based semi-supervised approaches (Yao *et al.*, 2019a; Liu *et al.*, 2020; Hu *et al.*, 2019c) have been proposed for text classification to better model the relations among words and documents in the corpus. These approaches typically construct a single heterogeneous graph for the whole corpus containing word nodes and document nodes, and connect the nodes with edges based on word co-occurrence and document-word relations (Yao *et al.*, 2019a; Liu *et al.*, 2020). Hu *et al.* (2019c) proposed to enrich the semantics of the short text with additional information (i.e., topics and entities), and constructed a Heterogeneous Information Network (HIN) containing document, topic and entity nodes with document-topic, document-entity and entity-entity edges based on several predefined rules. One limitation of semi-supervised text classification is its incapability of handling unseen documents in the testing phase. In order to

handle the inductive learning setting, some GNN-based approaches (Defferrard *et al.*, 2016; Huang *et al.*, 2019; Zhang *et al.*, 2020e) proposed to instead build an individual graph of unique words for each document by leveraging word similarity or co-occurrence between words within certain fixed-sized context window. In comparison to static graph construction, dynamic graph construction does not rely on domain-specific prior knowledge, and the graph structure can be jointly learned with the remaining learning modules of the system. Henaff *et al.* (2015) proposed to jointly learn a graph of unique words for each input text using a Gaussian kernel. Chen *et al.* (2020f) proposed to regard each word in text as a node in a graph, and dynamically build a graph for each document.

- **Graph Representation Learning** Early graph-based text classification approaches (Henaff *et al.*, 2015; Defferrard *et al.*, 2016) were motivated by extending CNNs to graph CNNs which can directly model graph-structured textual data. With the fast growth of the GNN research, recent work started to explore various GNN models for text classification including GCN (Yao *et al.*, 2019a; Chen *et al.*, 2020f), GGNN (Zhang *et al.*, 2020e) and message passing mechanism (MPM) (Huang *et al.*, 2019). Liu *et al.* (2020) introduced a TensorGCN which first performs intra-graph convolution propagation and then performs inter-graph convolution propagation. Hu *et al.* (2019c) proposed a Heterogeneous GAT (HGAT) based on a dual-level (i.e., node-level and type-level) attention mechanism.

- **Node Embedding Initialization** Node embedding initialization is critical for the GNN performance. Interestingly, Yao *et al.* (2019a) observed in their experiments that by using only one-hot representation, a vanilla GCN (Yao *et al.*, 2019a) without any external word embeddings or knowledge already outperformed state-of-the-art methods for text classification. Nevertheless, most GNN-based approaches (Defferrard *et al.*, 2016; Hu *et al.*, 2019c; Huang *et al.*, 2019; Liu *et al.*, 2020; Zhang *et al.*, 2020e) still use pre-trained word embeddings to initialize node embeddings. Chen *et al.* (2020f) further applied a BiLSTM to a sequence of word embeddings to capture the contextual information of text for node embedding initialization.

- **Special Techniques** As a common trick used in text classification, some GNN-based approaches removed stop words during preprocessing (Henaff *et al.*, 2015; Yao *et al.*, 2019a; Zhang *et al.*, 2020e).

Benchmarks and Evaluation Common benchmarks for evaluating text classification methods include 20NEWS (Lang, 1995), Ohsumed (Hersh *et al.*, 1994), Reuters (Lewis *et al.*, 2004), Movie Review (MR) (Pang and Lee, 2004), AGNews (Zhang *et al.*, 2015), Snippets (Phan *et al.*, 2008), TagMyNews (Vitale *et al.*, 2012), and Twitter[1]. Accuracy is the most common evaluation metric.

0.7.5 Text Matching

Background and Motivation Most existing text matching approaches operate by mapping each text into a latent embedding space via some neural networks such as CNNs (Hu *et al.*, 2014; Pang *et al.*, 2016) or RNNs (Wan *et al.*, 2016), and then computing the matching score based on the similarity between the text representations. In order to model the rich interactions between two texts at different granularity levels, sophisticated attention or matching components are often carefully designed (Lu and Li, 2013; Palangi *et al.*, 2016; Yang *et al.*, 2016). Recently, there are a few works (Chen *et al.*, 2020c; Liu *et al.*, 2019a) successfully exploring GNNs for modeling the complicated interactions between text elements in the text matching literature.

Methodologies In this section, we introduce and summarize some representative GNN-related techniques adopted in recent text matching approaches.

- **Graph Construction** Chinese short text matching heavily relies on the quality of word segmentation. Instead of segmenting each sentence into a word sequence during preprocessing which can be erroneous, ambiguous or inconsistent, Chen *et al.* (2020c) proposed to construct a word lattice graph from all possible segmentation paths. Specifically, the word lattice graph contains all character subsequences that match words in a lexicon as nodes, and adds an edge between two nodes if they are adjacent in the original sentence. As a result, the constructed

[1] https://www.nltk.org/

graph encodes multiple word segmentation hypotheses for text matching. In order to tackle long text matching, Liu *et al.* (2019a) proposed to organize documents into a graph of concepts (i.e., a keyword or a set of highly correlated keywords in a document), and built a concept interaction heterogeneous graph that consists of three types of edges including keyword-keyword, keyword-concept and sentence-concept edges. Specifically, they first constructed a keyword co-occurrence graph, and based on that, they grouped keywords into concepts by applying community detection algorithms on the keyword co-occurrence graph. Finally, they assigned each sentence to the most similar concept.

- **Graph Representation Learning** Liu *et al.* (2019a) applied a GCN model to learn meaningful node embeddings in the constructed graph. Chen *et al.* (2020c) designed a GNN-based graph matching module which allows bidirectional message passing across nodes in both text graphs. In order to obtain graph-level embeddings from the learned node embeddings, max pooling (Liu *et al.*, 2019a) or attentive pooling (Chen *et al.*, 2020c) techniques were adopted.

- **Node Embedding Initialization** As for node embedding initialization, Chen *et al.* (2020c) used the BERT embeddings while Liu *et al.* (2019a) first computed a match vector for each node in the graph.

Benchmarks and Evaluation Common benchmarks for text matching include LCQMC (Liu *et al.*, 2018c), BQ (Chen *et al.*, 2018a), CNSE (Liu *et al.*, 2019a), and CNSS (Liu *et al.*, 2019a). Accuracy and F1 are the most widely used evaluation metrics.

0.7.6 Topic Modeling

Background and Motivation The task of topic modeling aims to discover the abstract "topics" that emerge in a corpus. Typically, a topic model learns to represent a piece of text as a mixture of topics where a topic itself is represented as a mixture of words from a vocabulary. Classical topic models include graphical model based methods (Blei *et al.*, 2003; Blei *et al.*, 2010), autoregressive model based methods (Larochelle and Lauly, 2012), and autoencoder based methods (Miao *et al.*, 2016; Chen and Zaki, 2017; Isonuma

et al., 2020). Recent works (Zhu *et al.*, 2018; Zhou *et al.*, 2020a; Yang *et al.*, 2020) have explored GNN-based methods for topic modeling by explicitly modeling the relationships between documents and words.

Methodologies In this section, we introduce and summarize some representative GNN-related techniques adopted in recent topic modeling approaches.

- **Graph Construction** How to construct a high-quality graph which naturally captures useful relationships between documents and words is the most important for GNN applications in the topic modeling task. Various graph construction strategies have been proposed for GNN-based topic models. In order to explicitly model the word co-occurrence, Zhu *et al.* (2018) extracted the biterms (i.e., word pairs) within a fixed-length text window for every document from a sampled mini-corpus, and built an undirected biterm graph where each node represents a word, and each edge weight indicates the frequency of the corresponding biterm in the mini-corpus. Zhou *et al.* (2020a) built a graph containing documents and words in the corpus as nodes, and added edges to connect document nodes and word nodes based on co-occurrence information where the edge weight matrix is basically the TF-IDF matrix. Yang *et al.* (2020) converted a corpus to a bi-partite graph containing document nodes and word nodes, and the edge weight indicates the frequency of the word in the document.

- **Graph Representation Learning** Given a graph representation of the corpus, Zhu *et al.* (2018) designed a GCN-based autoencoder model to reconstruct the input biterm graph. In addition, residual connections were introduced to the GCN architecture so as to avoid oversmoothing when stacking many GCN layers. Similarly, Zhou *et al.* (2020a) designed a GCN-based autoencoder model to restore the original document representations. Notably, Zhu *et al.* (2018) and Zhou *et al.* (2020a) reused the adjacency matrix as the node feature matrix which captures the word co-occurrence information. During the inference time, for both of the autoencoder-based methods, the weight matrix of the decoder network can be interpreted as the (unnormalized) word distributions of the

learned topics. Given the observations that Probabilistic Latent Semantic Indexing (pLSI) (Hofmann, 1999) can be interpreted as stochastic block model (SBM) (Abbe, 2017) on a specific bi-partite graph, and GAT can be interpreted as the semi-amortized inference of SBM, Yang *et al.* (2020) proposed a GAT-based topic modeling network to model the topic structure of non-i.i.d documents. As for node embedding initialization, they used pre-trained word embeddings to initialize word node features, and term frequency vectors to initialize document node features.

Benchmarks and Evaluation Common benchmarks for the topic modeling task include 20NEWS (Lang, 1995), All News (Thompson, 2017), Grolier (Wang *et al.*, 2019d), NYTimes (Wang *et al.*, 2019d), and Reuters (Lewis *et al.*, 2004). As for evaluation metrics, since it is challenging to annotate the ground-truth topics for a document, topic coherence score and case study on learned topics are typical means of judging the quality of the learned topics. Besides, with the topic representations of text output by a topic model, the performance on downstream tasks such as text classification can also be used to evaluate topic models.

0.7.7 Sentiment Classification

Background and Motivation The sentiment classification task aims to detect the sentiment (i.e., positive, negative or neutral) of a piece of text (Pang *et al.*, 2002). Unlike general sentiment classification, aspect level sentiment classification aims at identifying the sentiment polarity of text regarding a specific aspect, and has received more attention (Pontiki *et al.*, 2014). While most works focus on sentence level and single domain sentiment classification, some attempts have been made on document level (Chen *et al.*, 2020e) and cross-domain (Ghosal *et al.*, 2020) sentiment classification. Early works on sentiment classification heavily relied on feature engineering (Jiang *et al.*, 2011). Recent attempts (Tang *et al.*, 2016; Huang and Carley, 2018) leveraged the expressive power of various neural network models such as LSTM (Hochreiter and Schmidhuber, 1997), CNN (LeCun and Bengio, 1998) or Memory Networks (Sukhbaatar *et al.*, 2015). Very recently, more attempts have been made to leverage GNNs to better model syntactic and semantic meanings of

text for the sentiment classification task.

Methodologies GNN-based sentiment classification approaches typically operate by first constructing a graph representation (e.g., dependency tree) of the text, and then applying a GNN to learn good text embeddings which will be used for predicting the sentiment polarity. The graph construction techniques and graph representation techniques developed for solving the sentiment classification task vary between different approaches. In this section, we introduce and summarize some representative GNN-related techniques adopted in recent sentiment classification approaches.

- **Graph Construction** Most GNN-based approaches (Zhang *et al.*, 2019a; Sun *et al.*, 2019c; Huang and Carley, 2019; Pouran Ben Veyseh *et al.*, 2020; Tang *et al.*, 2020a; Wang *et al.*, 2020b) for sentence level sentiment classification used a dependency tree structure to represent the input text. Besides using a dependency graph for capturing syntactic information, Zhang and Qian (2020) in addition constructed a global lexical graph to encode the corpus level word co-occurrence information, and further built a concept hierarchy on both the syntactic and lexical graphs. Ghosal *et al.* (2020) constructed a subgraph from ConceptNet (Speer *et al.*, 2017) using seed concepts extracted from text. To capture the document-level sentiment preference information, Chen *et al.* (2020e) built a bipartite graph with edges connecting sentence nodes to the corresponding aspect nodes for capturing the intra-aspect consistency, and a graph with edges connecting sentence nodes within the same document for capturing the inter-aspect tendency.

- **Graph Representation Learning** Both the design of GNN models and quality of initial node embeddings are critical for the overall performance of GNN-based sentiment classification methods. Common GNN models adopted in the sentiment classification task include GCN (Zhang *et al.*, 2019a; Sun *et al.*, 2019c; Pouran Ben Veyseh *et al.*, 2020; Zhang and Qian, 2020), GAT (Huang and Carley, 2019; Chen *et al.*, 2020e) and Graph Transformer (Tang *et al.*, 2020a). To handle multi-relational graphs, R-GCN (Ghosal *et al.*, 2020) and R-GAT (Wang *et al.*, 2020b) were also applied to perform relation-aware message passing over

graphs. Most approaches used GloVe+BiLSTM (Sun *et al.*, 2019c; Tang *et al.*, 2020a; Wang *et al.*, 2020b; Tang *et al.*, 2020a) or BERT (Huang and Carley, 2019; Pouran Ben Veyseh *et al.*, 2020; Chen *et al.*, 2020e; Wang *et al.*, 2020b; Tang *et al.*, 2020a) to initialize node embeddings.

- **Special Techniques** One common trick used in aspect level sentiment classification is to include position weights or embeddings (Zhang *et al.*, 2019a; Zhang and Qian, 2020) to emphasize more on tokens closer to the aspect phase.

Benchmarks and Evaluation Common benchmarks for evaluating sentiment classification methods include Twitter (Dong *et al.*, 2014), SemEval sentiment analysis datasets (Pontiki *et al.*, 2014; Pontiki *et al.*, 2015; Pontiki *et al.*, 2016), MAMS (Jiang *et al.*, 2019), and Amazon-reviews (Blitzer *et al.*, 2007). Accuracy and F1 are the most common evaluation metrics.

0.7.8 Knowledge Graph

Knowledge graph (KG), which represents the real world knowledge in a structured form, has attracted a lot of attention in academia and industry. KG can be denoted as a set of triples of the form $\langle subject, relation, object \rangle$. There are three main tasks in term of KG, namely, knowledge graph embedding (KGE), knowledge graph completion (KGC), and Knowledge Graph Alignment (KGA). KGE aims to map entities and relations into low-dimensional vectors, which usually regarded as the sub-task in KGC and KGA. In this section, we will give a overview of the graph-based approaches to KGC and KGA.

Knowledge Graph Completion

Background and Motivation The purpose of KGC is to predict new triples on the basis of existing triples, so as to further extend KGs. KGC is usually considered as a link prediction task. Formally, the knowledge graph is represented by $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$, in which entities $v_i \in \mathcal{V}$, edges $(v_s, r, v_o) \in \mathcal{E}$, and $r \in \mathcal{R}$ is a relation type. This task scores for new facts (i.e. triples like $\langle subject, relation, object \rangle$) to determine how likely those edges are to belong to $\mathcal{E}$.

Methodologies. KGC can be solved with an encoder-decoder framework. To encode the local neighborhood information of an entity, the encoder can be chosen from a variety of GNNs such as GCN(Malaviya *et al.*, 2020; Shang *et al.*, 2019), R-GCN (Schlichtkrull *et al.*, 2018; Teru *et al.*, 2020) and Attention-based GNNs (Nathani *et al.*, 2019b; Bansal *et al.*, 2019; Zhang *et al.*, 2020g; Wang *et al.*, 2019a). Then, the encoder maps each entity (subject entity and object entity) $v_i \in \mathcal{V}$ to a real-valued vector $e_i \in \mathbb{R}^d$. Relation can be represented as an embedding e_r or a matrix M_r. Following the framework concluded by Wang *et al.* (2019g), the GNN encoder in a multi-relational graph (such as KG) can be formulated as:

$$\begin{aligned} a_v^{(l)} &= AGGREGATE_l(h_{r,u}^{(l-1)}, \forall u \in \mathcal{N}_v^r) \\ h_v^{(l)} &= COMBINE_l(h_{r0,v}^{(l-1)}, a_v^{(l)}) \end{aligned} \tag{93}$$

where $h_{r,u}^{(l-1)}$ denotes the message passing from the neighbor node u under relation r at the l th layer. For example, RGCN (Schlichtkrull *et al.*, 2018) sets $h_{r,u}^{(l-1)} = W_r^{(l-1)} h_u^{(l-1)}$ and $AGGREGATE(\cdot)$ be mean pooling. Since the knowledge graph is very large, the update of the node representation Eq.93 is efficiently implemented by using sparse matrix multiplications to avoid explicit summation over neighborhoods in practice.

The decoder is a knowledge graph embedding model and can be regarded as a scoring function. The most common decoders of knowledge graph completion includes translation-based models (TransE (Bordes *et al.*, 2013)), tensor factorization based models (DistMult (Yang *et al.*, 2014), ComplEx(Trouillon *et al.*, 2016)) and neural network base models (ConvE (Dettmers *et al.*, 2018b)). In Table 4 , we summarize these common scoring functions following Ji *et al.* (2020). $\mathrm{Re}(\cdot)$ denotes the real part of a vector, $*$ denotes convolution operator, ω denotes convolution filters and $g(\cdot)$ is a non-linear function. For example, RGCN uses DistMult as a scoring function, and DistMult can perform well on the standard link prediction benchmarks when used alone. In DistMult, every relation r is represented by a diagonal matrix $M_r \in \mathbb{R}^{d \times d}$ and a triple is scored as $f(s, r, o) = e_s^T M_r e_o$.

At last, the model is trained with negative sampling, which randomly corrupts either the subject or the object of each positive example. To optimize KGC models, cross-entropy loss (Schlichtkrull *et al.*, 2018; Wang *et al.*, 2019g; Zhang *et al.*, 2020g; Malaviya *et al.*, 2020) and margin-based loss (Teru *et al.*,

Table 4: KGC Scoring Function.

Model	Ent. embed.	Rel. embed.	Scoring Function $f(s,r,o)$
DistMult (Schlichtkrull *et al.*, 2018) (Wang *et al.*, 2019g) (Bansal *et al.*, 2019)	$e_s, e_o \in \mathbb{R}^d$	$M_r \in \mathbb{R}^{d \times d}$	$e_s^T M_r e_o$
ComplEx (Wang *et al.*, 2019g)	$e_s, e_o \in \mathbb{C}^d$	$e_r \in \mathbb{C}^d$	$\text{Re}(e_r, e_s, \bar{e_t}) = \text{Re}(\sum_{k=1}^{K} e_r e_s \bar{e_t})$
ConvKB (Nathani *et al.*, 2019a)	$e_s, e_o \in \mathbb{R}^d$	$e_r \in \mathbb{R}^d$	$\text{concat}(\sigma([e_s, e_r, e_o] * \omega)) \cdot w$
ConvE (Wang *et al.*, 2019a)	$M_s \in \mathbb{R}^{d_w \times d_h}, e_o \in \mathbb{R}^d$	$M_r \in \mathbb{R}^{d_w \times d_h}$	$\sigma(vec(\sigma([M_s; M_r] * \omega))W)e_o$
Conv-TransE (Shang *et al.*, 2019) (Malaviya *et al.*, 2020)	$e_s, e_o \in \mathbb{R}^d$	$e_r \in \mathbb{R}^d$	$g(vec(M(e_s, e_r))W)e_o$

2020; Nathani *et al.*, 2019a) are common loss functions used for optimizing KGC models.

Benchmarks and Evaluation Common KGC benchmark datasets include FB15k-237 (Dettmers *et al.*, 2018a), WN18RR (Toutanova *et al.*, 2015), NELL-995 (Xiong *et al.*, 2017b) and Kinship (Lin *et al.*, 2018). Two commonly used evaluation metrics are *mean reciprocal rank* (MRR) and *Hits at* n (H@n), where n is usually 1, 3, or 10.

Knowledge Graph Alignment

Background and Motivation . KGA aims at finding corresponding nodes or edges referring to the same entity or relationship in different knowledge graphs. KGA, such as cross-lingual knowledge graphs alignment, is useful for constructing more complete and compact KGs. Let $G_1 = (\mathcal{V}_1, \mathcal{E}_1, \mathcal{R}_1)$ and $G_2 = (\mathcal{V}_2, \mathcal{E}_2, \mathcal{R}_2)$ be two different KGs, and $S = \{(v_{i_1}, v_{i_2}) | v_{i_1} \in \mathcal{V}_1, v_{i_2} \in \mathcal{V}_2\}$ be a set of pre-aligned entity pairs between G1 and G2. The core task of KGA is entity or relation alignment, which is defined as finding new entity or relation alignments based on the existing ones.

Methodologies. GNN-based KGA or entity alignment approaches mostly use GNN models to learn the representations of the entities and relations in different KGs. Then, entity/relation alignment can be performed by computing the distance between two entities/relations. GCN is widely used in (Wang

et al., 2018; Xu *et al.*, 2019a; Wu *et al.*, 2019b). To further capture the relation information existing in multi-relational KGs, Wu *et al.* (2019a) proposed a Relation-aware Dual-Graph Convolutional Network (RDGCN), which also applied a graph attention mechanism. Similarly, Ye *et al.* (2019) also introduced relation information by proposing a vectorized relational graph convolutional network (VR-GCN). Cao *et al.* (2019b) proposed a Multi-channel Graph Neural Network model (MuGNN) containing a KG self-attention module and a cross-KG attention module to encode two KGs via multiple channels. GAT is another common model, which is applied in (Li *et al.*, 2019; Sun *et al.*, 2020b; Wang *et al.*, 2020h). Moreover, Sun *et al.* (2020b), Wu *et al.* (2019a), and Wu *et al.* (2019b) also introduced a gating mechanism to control the aggregation of neighboring information.

Entity/relation alignments are predicted by the distance between the entity/relation embeddings. The distance measuring functions are mainly based on L1 norm (Ye *et al.*, 2019; Wu *et al.*, 2019a; Wang *et al.*, 2018; Wu *et al.*, 2019b), L2 norm (Cao *et al.*, 2019b; Li *et al.*, 2019; Sun *et al.*, 2020b), cosine similarity (Xu *et al.*, 2019a), and feed-forward neural network (Xu *et al.*, 2019a; Wang *et al.*, 2020h).

Benchmarks and Evaluation. Common KGA benchmarks datasets include $DBP15K$ (Sun *et al.*, 2017) and $DWY100K$ (Sun *et al.*, 2018b). $DBP15K$ contains three cross-lingual datasets: DBP_{ZH-EN} (Chinese to English), DBP_{JA-EN} (Japanese to English), and DBP_{FR-EN} (French to English). $DWY100K$ is composed of two large-scale cross-resource datasets: $DWY-WD$ (DBpedia to Wikidata) and $DWY-YG$ (DBpedia to YAGO3). Hits@N, which is calculated by measuring the proportion of correctly aligned entities/relations in the top N list, is used as evaluation metric to assess the performance of the models.

0.7.9 Information Extraction

Background and Motivation Information Extraction (IE) aims to extract entity pairs and their relationships of a given sentence or document. IE is a significant task because it contributes to the automatic knowledge graph construction from unstructured texts. With the success of deep neural networks, NN-based methods have been applied to information extraction. However,

these methods often ignore the non-local and non-sequential context information of the input text (Qian *et al.*, 2019). Furthermore, the prediction of overlapping relations, namely the relation prediction of pairs of entities sharing the same entities, cannot be solved properly (Fu *et al.*, 2019). To these ends, GNNs have been widely used to model the interaction between entities and relations in the text.

Methodologies Information extraction composed of two sub-tasks: named entity recognition (NER) and relation extraction (RE). NER predicts a label for each word in a sentence, which is often regarded as a sequence tagging task (Qian *et al.*, 2019). RE predicts a relation type for every pair of entities in the text. When the entities are annotated in the input text, the IE task degrades into an RE task (Sahu *et al.*, 2019; Christopoulou *et al.*, 2019; Guo *et al.*, 2019b; Zhu *et al.*, 2019a; Zhang *et al.*, 2019d; Zhang *et al.*, 2018c; Vashishth *et al.*, 2018; Zeng *et al.*, 2020; Song *et al.*, 2018e). GNN-based IE approaches typically operate via a pipeline approach. First, a text graph is constructed. Then, the entities are recognized and the relationships between entity pairs are predicted. Very recently, researchers starts to jointly learn the NER and RE to take advantage of the interaction between these two sub-tasks (Fu *et al.*, 2019; Luan *et al.*, 2019; Sun *et al.*, 2019a). Followings are the introduction of different GNN-based techniques.

- **Graph Construction** Most GNN-based information extraction methods design specific rules to construct static graphs. Because the input of IE task is usually a document containing multiple sentences, the nodes in the constructed graph can be words, entity spans and sentences and the corresponding edges are word-level edges, span-level edges and sentence-level edges. These nodes can be connected by syntactic dependency edges (Fu *et al.*, 2019; Guo *et al.*, 2019b; Zhang *et al.*, 2018c; Vashishth *et al.*, 2018; Song *et al.*, 2018e; Sahu *et al.*, 2019), co-reference edges (Luan *et al.*, 2019; Zeng *et al.*, 2020; Sahu *et al.*, 2019), re-occurrence edges (Qian *et al.*, 2019), co-occurrence edges (Christopoulou *et al.*, 2019; Zeng *et al.*, 2020), adjacent word edges (Qian *et al.*, 2019; Luan *et al.*, 2019; Sahu *et al.*, 2019) and adjacent sentence edge (Sahu *et al.*, 2019). Recently, dynamic graph construction has also been successfully applied in IE tasks. (Luan *et al.*, 2019) proposed

a general IE framework using dynamically constructed span graphs, which selected the most confident entity spans from the input document and linked these span nodes with co-references and confidence-weighted relation types. (Sun *et al.*, 2019a) first constructs a static entity-relation bipartite graph and then investigates the dynamic graph for pruning redundant edges.

- **Graph Representation Learning** To better capture non-local information of the input document, a variety of GNN models are applied in the NER task and the RE task. In addition, joint learning is a critical technique to reduce error propagation along the pipeline. For the name entity recognition task, common GNN models such as GCN (Qian *et al.*, 2019; Luo and Zhao, 2020) are applied. GCN is the most common GNN models used in the relation extraction task (Zhang *et al.*, 2019d; Zhang *et al.*, 2018c; Vashishth *et al.*, 2018; Zeng *et al.*, 2020). To learn edge type-specific representations, Sahu *et al.* (2019) introduces a labelled edge GCN to keep separate parameters for each edge type. Inspired by the graph attention mechanism, Guo *et al.* (2019b) proposes attention guided GCN to prune the irrelevant information from the dependency trees. Recently, many joint learning models have been proposed to relieve the error propagation in the pipeline IE systems and leverage the interaction between the NER task and the RE task. Fu *et al.* (2019) proposes a GraphRel model containing 2 phases prediction of the entities and relations. Luan *et al.* (2019) introduces a general framework to couple multiple information extraction sub-tasks by sharing entity span representations which are refined using contextualized information from relations and co-references. Sun *et al.* (2019a) develops a paradigm that first detected entity spans, and then performed a joint inference on entity types and relation types.

Benchmarks and Evaluation. Common IE benchmark datasets contain NYT (Riedel *et al.*, 2010), WebNLG (Gardent *et al.*, 2017), ACE2004, ACE2005, SciERC(Luan *et al.*, 2018), TACRED (Zhang *et al.*, 2017b) and etc. Precision, recall and F1 are the most common evaluation metrics for IE.

0.7.10 Semantic and Syntactic Parsing

In this section, we mainly discuss applications of GNN for parsing, including syntax related and semantics related parsing. For syntax related parsing, GNN has been employed in tasks of dependency parsing(Ji *et al.*, 2019)(Do and Rehbein, 2020) and constituency parsing(Yang and Deng, 2020). For semantics related parsing, we will briefly introduce semantic parsing and AMR (Abstract Meaning Representation) parsing.

Syntax Related Parsing

Background and motivation The tasks related to syntax are mainly dependency parsing and constituency parsing. Both of them aim to generate a tree with syntactic structure from natural language sentences, conforming to predefined formal grammar rules. Dependency parsing focuses on the dependency relationship between words in sentence. Constituency parsing focuses on the compositional relationship between different components in a sentence. Traditional approaches can be divided into two directions: transition-based and graph-based. Transition-based methods(Andor *et al.*, 2016)(Ma *et al.*, 2018) usually formalize this problem as a series of decisions on how to combine different words into a syntactic structure. Graph-based methods(Kiperwasser and Goldberg, 2016)(Dozat and Manning, 2016)(Ji *et al.*, 2019) firstly score all word pairs in a sentence on the possibility of holding valid dependency relationship, and then exploit decoders to generate the parse trees.

Methodologies Here, we mainly focus on graph-based parsers where graph neural network plays the role of extracting high-order neighbor features.

- **Dependency parsing**. In graph-based parsers, we take each word as a node in a graph and the key task is to learn a low-dimensional node representation with a neural encoder. To incorporate more dependency structure information, Ji *et al.* (2019) proposes to employ GNN as a encoder to incorporate high-order information. Their encoder contains both GNN and Bi-LSTM, where the GNN accepts all node embeddings from Bi-LSTM and take them as node embeddings in a complete graph. The constructed graphs are dynamic graphs where edge weight can change consistently during training. There are two kinds of loss

functions: 1) the first one considers both tree structure and dependency relation labels; 2) the second one are applied after each GNN layer where only tree structure is considered. Other than the generation of dependency parsing trees, some other works focus on how to do reranking among different candidates to choose a best parsing tree. Do and Rehbein (2020) demonstrate that GNN can also work well as a encoder for dependency parsing trees in a neural reranking model.

- **Constituency parsing**. Most approaches for constituency parsing are transition-based (Sagae and Lavie, 2005; Dyer *et al.*, 2016; Liu and Zhang, 2017; Yang and Deng, 2020) which generate the constituency parsing tree by executing an action sequences. Yang and Deng (2020) proposes to use GNN to encode the partial tree in the decoding process which can generate one token per step. Other methods usually generate the final parsing tree by combining different sub-trees in a shift-reduce way. The authors believe that this strongly incremental way is more closer to the way of human thinking.

Benchmark and Evaluation For syntactic parsing, two becnmark datasets are commonly used, namely, PTB 3.0(Taylor *et al.*, 2003) and UD 2.2(Nivre et al., 2018). As for evaluation, Accuracy, including exact match accuracy and execution accuracy, and Smatch score(Cai and Knight, 2013) are commanly used.

Semantics Related Parsing

Background and Motivation For semantics related tasks, we will introduce two popular applications: SQL parsing and AMR parsing. Semantic parsing aims to generate machine-interpretable representations from natural language, like SQL queries. AMR parsing is another young research field. AMR is represented as a rooted labeled directed acyclic graph form, and the goal of AMR parsing aims to provide sentence-level semantic representations. It is widely used in many NLP tasks like text summarization, machine translation and question answering(Zhou *et al.*, 2020b).

Methodologies Here, we provide a summary of the techniques for two typical semantic related parsing tasks, namely, SQL parsing and AMR parsing.

- **SQL parsing**. The main purpose of SQL parsing is to convert natural language into SQL queries that can be successfully executed. Most of the traditional methods (Jia and Liang, 2016; Alvarez-Melis and Jaakkola, 2016; Dong and Lapata, 2016) are sequential encoder based, which however, lost some other useful information at the source side, such as syntax information and DB schema information. Thus, many GNN-based models are proposed. For syntactic information, Li *et al.* (2020a) and Xu *et al.* (2018c) use external parser to perform syntactic parsing (i.e., constituency parsing and dependency parsing) on the raw sentence. Then they exploit the syntactic parsing tree instead of the source sentence as input, and use GNN to learn the syntactic structure and dependent information in this "tree" graph. It has been proved experimentally that the additional syntactic information is helpful for semantic parsing tasks. SQL parsing problem becomes more complex if the DB schema of the training and testing set are different (Yu *et al.*, 2018). To this end, some works propose to model these schema information to achieve better results. For example, Bogin *et al.* (2019a) takes the DB schema as a graph and use GGNN to learn the node representation. Then they incorporate schema information on both the encoder and decoder to generate the final results. Bogin *et al.* (2019b) employs GNN to globally select the overall structure of the output query which could decrease the ambiguity of DB constants choice.

  After the SQL queries are generated, reranking can be utilized to further improve the performance. Reranking the candidates predicted by the model is helpful to reduce the likelihood of picking some sub-optimal results. SQL queries are structured and it is a reasonable way to use GNN to encode the SQL queries in the reranking model. For example, Do and Rehbein (2020) employs graph-based transformer to rearrange the results generated by the neural semantic parser and achieved good results.

- **AMR parsing**. Similar to (Li *et al.*, 2020a)(Xu *et al.*, 2018c), syntactic information, especially dependency relation information, are also employed in AMR parsing. Zhou *et al.* (2020b) considers both the dependency syntactic graph and the latent probabilistic graph. Specifically, by learning a vector representation for the two graph structures and

then fusing them together, their model leverages the structure information in the source side and achieve better performance compared to seq-to-graph-like models.

Benchmark and Evaluation For SQL parsing, three benchmark datasets are commanly used, including ATIS(Dahl *et al.*, 1994), GEO(Luke, 2005), WikiSQL(Zhong *et al.*, 2017), SPIDER(Yu *et al.*, 2018). For AMR parsing, AMR annotation release(Knight *et al.*, 2014; Knight *et al.*, 2017) is a well-recognized dataset. For evaluation metrics, accuracy, including exact match accuracy and execution accuracy, as well as Smatch score(Cai and Knight, 2013).) are commonly used.

0.7.11 Reasoning

Reasoning is a significant research direction for NLP. In recent years, GNN begins to play an important role in NLP reasoning tasks, such as math word problem solving (Li *et al.*, 2020a; Zhang *et al.*, 2020b), natural language inference (Kapanipathi *et al.*, 2020; Wang *et al.*, 2019f), common sense reasoning (Lin *et al.*, 2019a; Zhou *et al.*, 2018a) and so on. In this subsection, we will give a brief introduction for the three tasks and how graph neural networks are employed in these methods.

Math word problem solving

Background and Motivation Math word problem solving aims to infer reasonable equations from given natural language problem descriptions. It is important for exploring automatic solutions to mathematical problems and improving the reasoning ability of neural networks. Most of the traditional methods are based on the seq2seq (Wang *et al.*, 2017) framework to generate the corresponding equation directly from the source sentence in an end-to-end manner. This kind of methods ignore some important information in natural sentences, such as 1) the relationship information between different mathematical elements (numbers) in the question, 2) the syntax information in the question sentence, 3) external knowledge, and so on. Thus, GNN-based models are proposed as a very good way to incorporate this information.

Methodologies Li *et al.* (2020a) is the first to introduce GNN into math word problem solving. Graph2tree considers both the input and output structure information. At the input side, GNN is used to encode the input syntactic tree. After all the input nodes embedding are generated, on the output side, considering the hierarchy of the equation, a BFS-based tree decoder is used to generate the final equation result in a coarse-to-fine way. Zhang *et al.* (2020b) is another MWP automatic solving model that uses graph data structure to model 1) the relationship between the numbers in the problem, and 2) the relationship between different numbers with their corresponding descriptors. In addition, some works introduce the external knowledge information in another way. For example, Wu *et al.* (2020b) first connects the entities in the problem description into graphs based on external global knowledge information, and then uses GAT as encoder. This method can enhance the ability of modeling the relationship between the entities in the problem, and has obtained good results.

Benchmarks and Evaluation For math word problem, three benchmark datasets are commonly used, including MAWPS (Koncel-Kedziorski *et al.*, 2016), MATH23K (Wang *et al.*, 2017), and MATHQA (Amini *et al.*, 2019).

Natural language inference

Background and Motivation Natural language inference (NLI) is another fundamental reasoning task. This task aims to predict the relationship between premise and hypothesis, and is often formalized as a three-classification problem (contradict, entails, neutral).

Methodologies Traditional methods are mostly based on neural encoder with attention, and most of them are RNN models(Chen *et al.*, 2017c). Considering the rich information contained in the external knowledge base, some works try to use external information to improve the accuracy of the model. For example, Wang *et al.* (2019f) uses graph-based attention model to incorporate the information from introduced external knowledge source. Their experiments demonstrate that adding the learned knowledge graph representation to the classifier help to obtain good results. Considering the introduced graph can have noisy information, Kapanipathi *et al.* (2020) employs a encoder with

a subgraph filtering module using Personalized PageRank before a GCN layer where the filtering module can help to select context relevant sub-graphs from introduced knowledge graph to reduce noisy information.

Benchmarks and Evaluation For NLI task, three benchmark datasets are commonly used, including SNLI(Bowman *et al.*, 2015), MultiNLI(Williams *et al.*, 2018), and SciTail(Khot *et al.*, 2018).

Commonsense reasoning

Background and Motivation Commonsense reasoning helps neural models incorporate the "common sense" or world knowledge during inference. Take the commonsense QA as example, we aim to obtain a neural model tended to generate the answer which is more consistent with commonsense from multiple answers that all logically fit the requirements. In fact, large-scale pre-trained models such as GPT-2(Radford *et al.*, 2019), BERT(Devlin *et al.*, 2019) with simple fine-tuning can achieve very good results. However, some external knowledge sources can help the model to better characterize the question and the concepts in the answer, which will definitely help the overall performance.

Methodologies Lin *et al.* (2019a) introduces graph neural networks to the common sense reasoning task. The model first retrieves the concepts in the questions and options into an external knowledge base to obtain a schema graph, and then uses GCN to incorporate information from this retrieved graph to learned features. The learnt features would be fed to a simple score module for each QA pair. Experiments on large benchmarks dataset, e.g., CommonsenseQA (Talmor *et al.*, 2019), demonstrate the effectiveness of the external knowledge base introduced by GNN.

Benchmarks and Evaluation We introduce some benchmark datasets for commonsense reasoning here: CommonsenseQA (Talmor *et al.*, 2019); Event2Mind(Rashkin *et al.*, 2018); SWAG(Zellers *et al.*, 2018); Winograd Schema Challenge(Levesque *et al.*, 2012); ReCoRD(Zhang *et al.*, 2018a).

0.7.12 Semantic Role Labelling

Background and Motivation The problem of semantic role labeling (SRL) aims to recover the predicate-argument structure of a sentence, namely, to determine essentially "who did what to whom", "when", and "where. More formally, for every predicate, the SRL model must identify all argument spans and label them with their semantic roles. Such high-level structures can be used as semantic information for supporting a variety of downstream tasks, including dialog systems, machine reading and translation (Shen and Lapata, 2007; Liu and Gildea, 2010; Gao and Vogel, 2011). Recent SRL works can mostly be divided into two categories, i.e., syntax-aware (Xia *et al.*, 2020; Marcheggiani and Titov, 2020) and syntax-agnostic (He *et al.*, 2017; He *et al.*, 2018) approaches according to whether incorporating syntactic knowledge or not. Most syntax-agnostic works employ deep BiLSTM or self-attention encoder to encode the contextual information of natural sentences, with various kinds of scorers to predict the probabilities of BIO-based semantic roles (He *et al.*, 2017) or predicate-argument-role tuples (He *et al.*, 2018). Motivated by the strong interplay between syntax and semantics, researchers explore various approaches to integrate syntactic knowledge into syntax-agnostic models considering that the semantic representations are closely related to syntactic ones. For example, one can observe that many arcs in the syntactic dependency graph are mirrored in the semantic dependency graph. Given these similarities and the availability of accurate syntactic parser for many languages, it seems natural to exploit syntactic information when predicting semantics.

However, the last generation of SRL models powered by deep learning models put syntax aside in favor of neural sequence models, namely LSTMs (Zhou *et al.*, 2020a; Marcheggiani *et al.*, 2017) due to the challenges that (1) it is difficult to effectively incorporate syntactic information into neural SRL models, due to the sophisticated tree structure of syntactic relation; and (2) the syntactic parsers are unreliable on account of the risk of erroneous syntactic input, which may lead to error propagation and an unsatisfactory SRL performance. Given this situation, GNNs are emerging as powerful tools to capture and incorporate the syntax patterns into deep neural network-based SRL models. The nature property of GNN in capturing the complex relationship patterns in the structured data makes it a good fit for modeling syntactic dependency and constituency structures of sentences.

Methodologies The problem solved by the GNN-based SRL models can be divided into two categories. One is about argument prediction given the predicates in a sentence (Marcheggiani and Titov, 2020; Marcheggiani and Titov, 2017; Li *et al.*, 2018c). Formally, SRL can be cast as a sequence labeling problem where given an input sentence, and the position of the predicate in the sentence, the goal is to predict a BIO sequence of semantic roles for the words in sentences; Another is about end-to-end semantic role triple extraction which aims to detect all the possible predicates and their corresponding arguments in one shot (Fei *et al.*, 2020; Xia *et al.*, 2020). Technically, given a sentence, the SRL model predicts a set of labeled predicate-argument-role triplets, while each triple contains a possible predicate token and two candidate tokens. Both of the above mentioned problems can be solved based on the GNN-based SRL models, which consists of two parts, namely, graph construction and graph representation learning.

- **Graph Construction**. The graphs are constructed based on the syntax information, which can be extracted from two sources, one is syntactic dependency information and another is syntactic constituents information. Most of the existing GNN-SRL models (Li *et al.*, 2018c; Fei *et al.*, 2020; Marcheggiani and Titov, 2017; Zhang *et al.*, 2020a; Xia *et al.*, 2020) have relied on syntactic dependency representations. In these methods, information from dependency trees are injected into word representations using GNN or self-attention mechanisms. Recently, Marcheggiani et al (Marcheggiani and Titov, 2020) incorporated the constituency syntax into SRL models by conducting the message passing on a graph where nodes represent constituents. Based on the syntax information, the graphs constructed in the current SRL models are divided into three main categories: (1) directed homogeneous graphs; (2) heterogeneous graphs; and (3) probability weighted graphs. Most of the works (Marcheggiani and Titov, 2017; Li *et al.*, 2018c; Fei *et al.*, 2020) represent the syntax information as a directed homogeneous graph where all the nodes are input word tokens and directed with dependent edges. Other work (Xia *et al.*, 2020) enhances SRL with heterogeneous syntactic knowledge by combining various syntactic treebanks that follow different annotation guidelines and domains. Liu et al. (Liu *et al.*, 2019c) also construct a heterogeneous syntactic graph by incorporating

several types of edges, including lexical relationships, syntactic dependency, co-occurrence relationships. Some work (Zhang *et al.*, 2020a) utilizes the probability matrix of all dependency arcs for constructing an edge-weighted directed graph to eliminate the influences of the error from the parsing results.

- **Graph Representation Learning**. As described in Section 6, various GNN models can be utilized for graph representation learning. Here, we introduce the different roles that GNNs play in different SRL models. In most of the works (Zhang *et al.*, 2020a; Liu *et al.*, 2019c; Marcheggiani and Titov, 2017; Marcheggiani and Titov, 2020), GNN is utilized as an encoder to learn the final representations of words which follows a typical word embedding layer, such as BiLSTM. While in some works (Xia *et al.*, 2020; Fei *et al.*, 2020), GNN is utilized to extract the initial words' embedding, which are regarded as inputs of the encoder. For example, Xia et al. (Xia *et al.*, 2020) combines the syntax embedding extracted from GNN with the word embedding and character embedding as the input. Fei (Fei *et al.*, 2020) utilizes GNN to refine the initial word embedding which consists of word representation and part-of-speech (POS) tags, and then input the refined word embedding into the BiLSTM encoder.

Benchmarks and Evaluation There are two main benchmark datasets for the evaluation in the domain of SRL: (1) CoNLL dataset concerns the recognition of semantic roles for the English language, based on PropBank predicate-argument structures. Given a sentence, the task consists of analyzing the propositions expressed by some target verbs of the sentence. In particular, for each target verb all the constituents in the sentence which fill a semantic role of the verb have to be recognized. (2) Chinese Proposition Bank 1.0 (CPB1.0) which creates a corpus of text annotated with information about basic semantic propositions (i.e., predicate-argument relations). The typical evaluation metrics in SLR task are about metrics for classification problem, such as precision, recall and F1 of the correctly predicted arguments.

0.7.13 Related Libraries and Codes

Open-source implementations facilitate the research works of baseline experiments in graph neural networks for NLP. Besides various paper codes were released individually, there is a recently released library called *Graph4NLP* [2], which is an easy-to-use library for R&D at the intersection of Deep Learning on Graphs and Natural Language Processing. It provides both full implementations of state-of-the-art models mentioned above for several NLP applications including text classification, semantic parsing, machine translation, KG completion, and natural language generation. *Graph4NLP* also provides flexible interfaces to build customized models for researchers and developers with whole-pipeline support. Built upon highly-optimized runtime libraries including *DGL* and *Pytorch*, Graph4NLP has both high running efficiency and great extensibility. The architecture of *Graph4NLP* consists of four different layers: 1) Data Layer, 2) Module Layer, 3) Model Layer, and 4) Application Layer. There are also some other related GNN-based libraries. Noticeably, Fey and Lenssen (2019) published a geometric learning library in *PyTorch* named *PyTorch Geometric*, which implements many GNNs. The *Deep Graph Library (DGL)* (Wang *et al.*, 2019b) was released which provides a fast implementation of many GNNs on top of popular deep learning platforms such as *PyTorch* and *MXNet*. The *Dive into Graphs* (Liu *et al.*, 2021a) was released recently as a research-oriented library that integrates unified and extensible implementations of common graph deep learning algorithms for several advanced tasks.

0.8 General Challenges and Future Directions

In this chapter, we will discuss various general challenges of GNNs for NLP and pinpoint the future research directions. We believe putting more research efforts in these directions will further unleash the great potential of GNNs in the NLP field, and result in fruitful research outcomes.

[2]The codes and details of *Graph4NLP* library are provided at https://github.com/graph4ai/graph4nlp.

0.8.1 Dynamic Graph Construction

As we see in Section 0.7, many NLP problems can be tackled from a graph perspective, and GNNs are naturally applicable to and good at handling graph-structured data. Thus, the graph construction process plays an important role in the overall model performance. However, constructing a high quality and task-specific graph requires a good amount of domain expertise and human effort. Moreover, graph construction in NLP is often more art than science, informed solely by insight of the practitioner, and involves many trials and errors. Even though a few of existing works already explored dynamic graph construction, most GNN applications in NLP still heavily relied on domain expertise for static graph construction.

The exploration of dynamic graph construction for NLP, is still at its early stage and faces several challenges: first of all, most works on dynamic graph construction focused only on homogeneous graph construction (Chen *et al.*, 2020i; Chen *et al.*, 2020h; Chen *et al.*, 2020f; Liu *et al.*, 2021b; Liu *et al.*, 2019c), and dynamic graph construction for heterogeneous graphs (Yun *et al.*, 2019; Zhao *et al.*, 2021) is much less explored especially for NLP tasks. Compared to homogeneous graphs, heterogeneous graphs are capable of carrying on richer information on node types and edge types, and occur frequently in many NLP problems. Dynamic graph construction for heterogeneous graphs is also supposed to be more challenging because more types of information (e.g., node types, edge types) are expected to be learned from data.

Second, most existing dynamic graph construction techniques rely on some form of pair-wise node similarity computation whose time complexity is at least $O(n^2)$ where n is the number of graph nodes. This results in scalability issues when scaling to large graphs such as KGs. Recently, a scalable graph structure learning approach with linear time and memory complexity (in terms of the number of nodes) was proposed by adopting the anchor-based approximation technique to avoid explicit pair-wise node similarity computation (Chen *et al.*, 2020f).

Finally, various efficient transformers (Tsai *et al.*, 2019; Katharopoulos *et al.*, 2020; Choromanski *et al.*, 2021; Peng *et al.*, 2021; Shen *et al.*, 2021; Wang *et al.*, 2020e) were also developed which could inspire the research in scalable dynamic graph construction considering their close connections; (3) As observed in some previous works, dynamic graph construction does not

clearly outperform static graph construction in some NLP applications. There is still room for improvement for dynamic graph construction techniques in terms of downstream task performance. Other interesting future directions in this line include dynamically learning edge directions for graphs, and combining static and dynamic graph construction for better performance.

0.8.2 GNNs vs Transformers for NLP

While GNNs have achieved promising results in a large variety of NLP fields, Transformers have received much more attentions due to excellent performance in many NLP applications. However, as we pointed out in the previous section, Transformers are type of special GNNs, which operated on a fully connected dynamic graph constructed by employing self-attention mechanism. Since both of them have their clear advantages over each other, there are several interesting direction worth exploiting here.

Combining GNNs with Transformers for NLP The most beautiful thing about Transformers is its simple use of its elegant model architecture directly on original text sequence, which separates the graph data modeling inside transformer model from the inputs (and from the end user). However, the downside of this model choice is that the Transformers cannot directly operate on more complex data like graph-structured data directly. In contrast, GNNs are more generic model architectures directly operating on graph data, which however are needed to be created by the end user with either domain specific knowledge or other graph modeling techniques. Currently, Graph Transformers are most popular models that adapt structure-aware self-attention mechanism to combine the advantages of Transformers and GNNs. However, these approaches are purely replying on attention mechanism to utilize the original graph topology information, which may not the best way to explore the original graph input information, especially when graph inputs are multi-relational and heterogeneous graphs.

Pre-training GNNs for NLP One of the most important trends in NLP over the past few years is to develop large-scale pre-trained models (Devlin *et al.*, 2018; Brown *et al.*, 2020) most of which are based on Transformer architectures. Recently, there are also many research efforts on pre-training

GNNs on graphs (Hu *et al.*, 2019d; Qiu *et al.*, 2020) using self-supervised learning methods. However, there are very few attempts to pre-train GNNs for NLP (He *et al.*, 2020; Sun *et al.*, 2020a; Chen *et al.*, 2020d), which may exploit more types of data sources than Transformers since GNNs can take both graph structured data (e.g., from KGs) and unstructured data (e.g., from free-form text).

0.8.3 Graph-to-Graph for NLP

Since the natural language or information knowledge can be naturally formalized as graphs with a set of nodes and their relationships, many generation tasks in the domain of NLP can be formalized as a graph transformation problem, which can further be solved by the graph-to-graph models. In this way, the semantic structural information of both the input and output sentences can be fully utilized and captured. For example, the graph-to-graph transformation for AMR parsing is promising and yet unexplored. Because AMR are naturally structured objects (e.g. tree structures), currently, many semantic AMR parsing methods are based on deep graph generative models. These methods (Flanigan *et al.*, 2014; Zhang *et al.*, 2019f; Cai and Lam, 2020a) represent the semantics of a sentence as a semantic graph (i.e., a sub-graph of a knowledge base) and treat semantic parsing as a semantic graph matching/generation process. These end-to-end deep graph generation techniques for semantic parsing show powerful ability in automatically capturing semantic information. However, current graph-based semantic parsing models only represent either the output AMR or the input sentence as a graph, without jointly utilize the interactive relationship in both input and output, and thus can not consider the complex relationship among the topology pattern of AMR logits and input dependency/constituency parsing.

While utilizing the graph-to-graph model for NLP tasks, there are several general challenges that deserve to be explored and solved in this domain: (1) Difficulty in building an end-to-end graph transformation framework which can integrate several sub-tasks jointly. For example, it is important to jointly tackle name entity recognition and relation extraction jointly in the information extraction; (2) The different concepts of input and output graphs. In NLP tasks, we usually formalize the dependency tree as the input graph and the output graph is usually has different concept from the input graph. Thus, both the node

set, and topology of the input and output graphs are different. For example, in AMR parsing, the nodes in output graph are AMR logits, while the nodes in input graphs are word tokens; (3) Difficulty in addressing the graph sparsity issue in NLP tasks. For example, AMR parsing is a much harder task in that the target vocabulary size is much larger, while the size of dataset is much smaller; (4) Unpaired graph-to-graph transformation. The annotations are relatively expensive to produce and thus corpora have on the order of tens of thousands of sentences. Utilizing the unpaired sample pairs is also important yet challenging.

0.8.4 Knowledge Graph in NLP

Knowledge graph has become an important component in many NLP tasks, such as question answering, natural language generation, knowledge graph completion and alignment. It can be either incorporated as an auxiliary information besides the input text to provide more knowledge (e.g., KG augmentation), or as an target object to be learnt or extracted from (e.g., KGE and KGC).

Knowledge Graph Augmentation For many NLP tasks such as QA and NLG, it is increasingly common to find that the input data not only contain Query text or source text, but also incorporate KG as auxiliary information for additional knowledge. There are several challenges with KG-augmented tasks: (1) There may exist ambiguity when aligning entities in the input text to entities in KG in some text-to-text generation tasks. For example, a person name appearing in the input text can correspond to multiple entries in KG; (2) Since the scale of KG is often large, it takes considerable effort to extract useful information from the KG and build knowledge subgraphs which can be directly used by every sample as an auxiliary input. Generally, detailed and task-specific rules need to be designed for KG node selection; (3) Entity alignment and knowledge subgraph construction may involve inevitable errors propagating to the downstream tasks. To make better use of KGs, the techniques for pre-processing KG, such as entity alignment and related entity/node selection, need to be further explored and improved.

Knowledge Graph Embedding and Completion GNN-based KGE and KGC approaches consider incorporating the neighborhood of a head entity

or a tail entity. There is a trade-off between *all triples training on the same large KG (Shang* et al.*, 2019)* and *each triple training on a separate knowledge subgraph constructed from the original KG (Teru* et al.*, 2020; Xie* et al.*, 2020)*. The former one provides more computational efficiency while the latter one has more powerful model expressiveness. Future research will focus on jointly reasoning text and KG by applying such methods and paying attention to the entities mentioned in the text (Bansal *et al.*, 2019). Logic rules play an important role to determine whether a triple is valid or not, which may be useful for KGE and KGC (Xie *et al.*, 2020).

Knowledge Graph Alignment Most of the existing GNN-based KG alignment models also face three critical challenges to be further explored: (1) Different KGs usually have heterogeneous schemas, and may mislead the representation learning , which makes it is difficult to integrate knowledge from different KGs (Cao *et al.*, 2019b; Wu *et al.*, 2019b); (2) The data in KG is usually incomplete (Sun *et al.*, 2020b) which needs pre-processing; (3) The seed alignments are limited (Li *et al.*, 2019). How to iteratively discover new entity alignments in the GNN-based framework is a future direction (Wang *et al.*, 2018; Li *et al.*, 2019).

0.8.5 Multi-relational Graph Neural Networks

Multi-relational graphs, which adopt unified nodes but relation-specific edges, are widely observed and explored in many NLP tasks. As we discussed in Section 0.5.2, most multi-relational GNNs, which are capable of exploiting multi-relational graphs, are extended from conventional homogeneous GNNs. Technically, most of them either apply relation-specific parameters during neighbor aggregation or split the heterogeneous graphs to homogeneous sub-graphs (Schlichtkrull *et al.*, 2018; Beck *et al.*, 2018a). Although impressive progresses have been made, there is still a challenge in handling over-parameterization problem due to the diverse relations existing in the graph. Although several tricks such as parameter-sharing (e.g, see Directed-GCN (Marcheggiani and Titov, 2017)) and matrix-decomposition (e.g., see R-GCN (Schlichtkrull *et al.*, 2018)) are widely used to enhance the models' generalization ability to address this issue, they still have limitations, such as resulting in the potential loss of the models' expression ability. There is a hard

trade-off between the over-parameterization and powerful model expression ability.

It is worth noting that various graph transformers have been introduced to exploit the multi-relational graphs (Yao *et al.*, 2020; Wang *et al.*, 2020f). However, the challenge exists in how to fully take advantage of the strong inductive bias (i.e., the graph topology) of the graphs by the transformers which are naturally free of that. Currently, most of them simply regard the self-attention's map as a fully-directed graph. On top of that, researchers either apply sparsing mechanisms (Yao *et al.*, 2020) or allow remote connection (Shaw *et al.*, 2018; Cai and Lam, 2020b) according to the given graph. How to develop an effective and general architecture for multi-relational graphs (or heterogeneous graphs) needs further exploration.

0.9 Conclusions

In this survey, we conduct a comprehensive overview of various graph neural networks evolving various NLP problems. Specifically, we first provide the preliminary knowledge of typical GNN models including graph filters and graph poolings. Then we propose a new taxonomy that systematically organizes GNNs for NLP approaches along three dimensions, namely, graph construction, graph representation learning, and the overall encoder-decoder models. Given these specific techniques at each stage of the NLP application pipelines, we discuss a wide range of NLP applications from the perspective of graph construction, graph representation learning, and special techniques. Finally, the general challenges and future directions in this line are provided to further unleash the great potential of GNNs in the NLP field.

References

Abbe, E. (2017). "Community detection and stochastic block models: recent developments". *The Journal of Machine Learning Research*. 18(1): 6446–6531.

Allahyari, M., S. Pouriyeh, M. Assefi, S. Safaei, E. D. Trippe, J. B. Gutierrez, and K. Kochut. (2017). "Text summarization techniques: a brief survey". *arXiv preprint arXiv:1707.02268*.

Allamanis, M., M. Brockschmidt, and M. Khademi. (2018). "Learning to Represent Programs with Graphs". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=BJOFETxR-.

Alon, U., S. Brody, O. Levy, and E. Yahav. (2018). "code2seq: Generating sequences from structured representations of code". *arXiv preprint arXiv:1808.01400*.

Alvarez-Melis, D. and T. S. Jaakkola. (2016). "Tree-structured decoding with doubly-recurrent neural networks".

Amini, A., S. Gabriel, P. Lin, R. Koncel-Kedziorski, Y. Choi, and H. Hajishirzi. (2019). "Mathqa: Towards interpretable math word problem solving with operation-based formalisms". *arXiv preprint arXiv:1905.13319*.

Andor, D., C. Alberti, D. Weiss, A. Severyn, A. Presta, K. Ganchev, S. Petrov, and M. Collins. (2016). "Globally Normalized Transition-Based Neural Networks". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics. 2442–2452. DOI: 10.18653/v1/P16-1231. URL: https://www.aclweb.org/anthology/P16-1231.

Angeli, G., M. J. Johnson Premkumar, and C. D. Manning. (2015). "Leveraging Linguistic Structure For Open Domain Information Extraction". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics. 344–354. DOI: 10.3115/v1/P15-1034. URL: https://www.aclweb.org/anthology/P15-1034.

Atwood, J. and D. Towsley. (2016). "Diffusion-Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Vol. 29. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2016/file/390e982518a50e280d8e2b535462ec1f-Paper.pdf.

Ba, J. L., J. R. Kiros, and G. E. Hinton. (2016). "Layer normalization". *arXiv preprint arXiv:1607.06450*.

Bahdanau, D., K. Cho, and Y. Bengio. (2015). "Neural Machine Translation by Jointly Learning to Align and Translate". In: *3rd International Conference on Learning Representations*. Ed. by Y. Bengio and Y. LeCun.

Bai, X., Y. Chen, L. Song, and Y. Zhang. (2021). "Semantic Representation for Dialogue Modeling". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 4430–4445.

Bai, X., L. Song, and Y. Zhang. (2020). "Online Back-Parsing for AMR-to-Text Generation". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics. 1206–1219. DOI: 10.18653/v1/2020.emnlp-main.92. URL: https://www.aclweb.org/anthology/2020.emnlp-main.92.

Bansal, T., D.-C. Juan, S. Ravi, and A. McCallum. (2019). "A2N: attending to neighbors for knowledge graph inference". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 4387–4392.

Bao, J., D. Tang, N. Duan, Z. Yan, Y. Lv, M. Zhou, and T. Zhao. (2018). "Table-to-text: Describing table region with natural language". In: *Thirty-Second AAAI Conference on Artificial Intelligence*.

Barone, A. V. M. and R. Sennrich. (2017). "A parallel corpus of Python functions and documentation strings for automated code documentation and code generation". *arXiv preprint arXiv:1707.02275*.

Bastings, J., I. Titov, W. Aziz, D. Marcheggiani, and K. Sima'an. (2017). "Graph Convolutional Encoders for Syntax-aware Neural Machine Translation". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics. 1957–1967. DOI: 10.18653/v1/D17-1209. URL: https://www.aclweb.org/anthology/D17-1209.

Beck, D., G. Haffari, and T. Cohn. (2018a). "Graph-to-Sequence Learning using Gated Graph Neural Networks". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics. 273–283. DOI: 10.18653/v1/P18-1026. URL: https://www.aclweb.org/anthology/P18-1026.

Beck, D., G. Haffari, and T. Cohn. (2018b). "Graph-to-Sequence Learning using Gated Graph Neural Networks". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 273–283.

Bengio, S., O. Vinyals, N. Jaitly, and N. Shazeer. (2015). "Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks". In: *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. Ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett. 1171–1179. URL: https://proceedings.neurips.cc/paper/2015/hash/e995f98d56967d946471af29d7bf99f1-Abstract.html.

Blei, D. M., T. L. Griffiths, and M. I. Jordan. (2010). "The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies". *Journal of the ACM (JACM)*. 57(2): 1–30.

Blei, D. M., A. Y. Ng, and M. I. Jordan. (2003). "Latent dirichlet allocation". *the Journal of machine Learning research*. 3: 993–1022.

Blitzer, J., M. Dredze, and F. Pereira. (2007). "Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification". In: *Proceedings of the 45th annual meeting of the association of computational linguistics*. 440–447.

Bogin, B., J. Berant, and M. Gardner. (2019a). "Representing Schema Structure with Graph Neural Networks for Text-to-SQL Parsing". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics. 4560–4565. DOI: 10.18653/v1/P19-1448. URL: https://www.aclweb.org/anthology/P19-1448.

Bogin, B., M. Gardner, and J. Berant. (2019b). "Global Reasoning over Database Structures for Text-to-SQL Parsing". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics. 3659–3664. DOI: 10.18653/v1/D19-1378. URL: https://www.aclweb.org/anthology/D19-1378.

Bordes, A., N. Usunier, S. Chopra, and J. Weston. (2015). "Large-scale simple question answering with memory networks". *arXiv preprint arXiv:1506.02075*.

Bordes, A., N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. (2013). "Translating embeddings for modeling multi-relational data". In: *Neural Information Processing Systems (NIPS)*. 1–9.

Bowman, S. R., G. Angeli, C. Potts, and C. D. Manning. (2015). "A large annotated corpus for learning natural language inference". *arXiv preprint arXiv:1508.05326*.

Brown, T. B., B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.* (2020). "Language models are few-shot learners". *arXiv preprint arXiv:2005.14165*.

Budzianowski, P., T.-H. Wen, B.-H. Tseng, I. Casanueva, S. Ultes, O. Ramadan, and M. Gasic. (2018). "MultiWOZ-A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling". In: *EMNLP*.

Cai, D. and W. Lam. (2020a). "AMR Parsing via Graph-Sequence Iterative Inference". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics. 1290–1301. DOI: 10.18653/v1/2020.acl-main.119. URL: https://www.aclweb.org/anthology/2020.acl-main.119.

Cai, D. and W. Lam. (2020b). "Graph Transformer for Graph-to-Sequence Learning". In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence*. AAAI Press. 7464–7471.

Cai, D. and W. Lam. (2020c). "Graph Transformer for Graph-to-Sequence Learning". *Proceedings of the AAAI Conference on Artificial Intelligence*. 34(05): 7464–7471. DOI: 10.1609/aaai.v34i05.6243. URL: https://ojs.aaai.org/index.php/AAAI/article/view/6243.

Cai, S. and K. Knight. (2013). "Smatch: an Evaluation Metric for Semantic Feature Structures". In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Sofia, Bulgaria: Association for Computational Linguistics. 748–752. URL: https://www.aclweb.org/anthology/P13-2131.

Campos, J. A., A. Otegi, A. Soroa, J. Deriu, M. Cieliebak, and E. Agirre. (2019). "Conversational qa for faqs". In: *3rd Conversational AI: "Today's Practice and Tomorrow's Potential" workshop*.

Cao, N. D., W. Aziz, and I. Titov. (2019a). "Question Answering by Reasoning Across Documents with Graph Convolutional Networks". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Ed. by J. Burstein, C. Doran, and T. Solorio. Association for Computational Linguistics. 2306–2317.

Cao, Y., Z. Liu, C. Li, J. Li, and T.-S. Chua. (2019b). "Multi-Channel Graph Neural Network for Entity Alignment". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 1452–1461.

Cao, Y., M. Fang, and D. Tao. (2019c). "BAG: Bi-directional Attention Entity Graph Convolutional Network for Multi-hop Reasoning Question Answering". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics. 357–362.

Casanueva, I., P. Budzianowski, P.-H. Su, N. Mrkšic, T.-H. Wen, S. Ultes, L. Rojas-Barahona, S. Young, and M. Gašic. (2017). "A Benchmarking Environment for Reinforcement Learning Based Task Oriented Dialogue Management". *stat*. 1050: 29.

Chen, C., Z. Teng, and Y. Zhang. (2020a). "Inducing target-specific latent structures for aspect sentiment classification". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 5596–5607.

Chen, D., A. Fisch, J. Weston, and A. Bordes. (2017a). "Reading wikipedia to answer open-domain questions". *arXiv preprint arXiv:1704.00051*.

Chen, H., X. Liu, D. Yin, and J. Tang. (2017b). "A survey on dialogue systems: Recent advances and new frontiers". *Acm Sigkdd Explorations Newsletter*. 19(2): 25–35.

Chen, J., Q. Chen, X. Liu, H. Yang, D. Lu, and B. Tang. (2018a). "The bq corpus: A large-scale domain-specific chinese corpus for sentence semantic equivalence identification". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 4946–4951.

Chen, L., B. Lv, C. Wang, S. Zhu, B. Tan, and K. Yu. (2020b). "Schema-Guided Multi-Domain Dialogue State Tracking with Graph Attention Neural Networks". In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press. 7521–7528.

Chen, L., B. Tan, S. Long, and K. Yu. (2018b). "Structured Dialogue Policy with Graph Neural Networks". In: *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*. Ed. by E. M. Bender, L. Derczynski, and P. Isabelle. Association for Computational Linguistics. 1257–1268.

Chen, L., Y. Zhao, B. Lyu, L. Jin, Z. Chen, S. Zhu, and K. Yu. (2020c). "Neural Graph Matching Networks for Chinese Short Text Matching". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*. Ed. by D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault. Association for Computational Linguistics. 6152–6158.

Chen, Q., X. Zhu, Z.-H. Ling, D. Inkpen, and S. Wei. (2017c). "Neural natural language inference models enhanced with external knowledge". *arXiv preprint arXiv:1711.04289*.

Chen, W., Y. Su, X. Yan, and W. Y. Wang. (2020d). "KGPT: Knowledge-Grounded Pre-Training for Data-to-Text Generation". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*. Association for Computational Linguistics. 8635–8648.

Chen, X., C. Sun, J. Wang, S. Li, L. Si, M. Zhang, and G. Zhou. (2020e). "Aspect Sentiment Classification with Document-level Sentiment Preference Modeling". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics. 3667–3677. DOI: 10.18653/v1/2020.acl-main.338. URL: https://www.aclweb.org/anthology/2020.acl-main.338.

Chen, Y., L. Wu, and M. J. Zaki. (2019). "Bidirectional Attentive Memory Networks for Question Answering over Knowledge Bases". In: *NAACL-HLT (1)*.

Chen, Y., L. Wu, and M. J. Zaki. (2020f). "Iterative Deep Graph Learning for Graph Neural Networks: Better and Robust Node Embeddings". In: *Proceedings of the 34th Conference on Neural Information Processing Systems*.

Chen, Y., L. Wu, and M. J. Zaki. (2020g). "Toward Subgraph Guided Knowledge Graph Question Generation with Graph Neural Networks". *arXiv preprint arXiv:2004.06015*.

Chen, Y., L. Wu, and M. J. Zaki. (2020h). "GraphFlow: Exploiting Conversation Flow with Graph Neural Networks for Conversational Machine Comprehension". In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*. ijcai.org. 1230–1236.

Chen, Y., L. Wu, and M. J. Zaki. (2020i). "Reinforcement Learning Based Graph-to-Sequence Model for Natural Question Generation". In: *Proceedings of the 8th International Conference on Learning Representations*.

Chen, Y. and M. J. Zaki. (2017). "Kate: K-competitive autoencoder for text". In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 85–94.

Cho, K., B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. (2014). "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. Ed. by A. Moschitti, B. Pang, and W. Daelemans. ACL. 1724–1734. DOI: 10.3115/v1/d14-1179. URL: https://doi.org/10.3115/v1/d14-1179.

Choi, E., H. He, M. Iyyer, M. Yatskar, W. T. Yih, Y. Choi, P. Liang, and L. Zettlemoyer. (2020). "QUAC: Question answering in context". In: *2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018*. Association for Computational Linguistics. 2174–2184.

Choromanski, K., V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlós, P. Hawkins, J. Davis, A. Mohiuddin, L. Kaiser, D. Belanger, L. Colwell, and A. Weller. (2021). "Rethinking Attention with Performers". In: *International Conference on Learning Representations*.

Christensen, J., S. Soderland, O. Etzioni, *et al.* (2013). "Towards coherent multi-document summarization". In: *Proceedings of the 2013 conference of the North American chapter of the association for computational linguistics: Human language technologies*. 1163–1173.

Christopoulou, F., M. Miwa, and S. Ananiadou. (2019). "Connecting the Dots: Document-level Neural Relation Extraction with Edge-oriented Graphs". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics. 4925–4936. DOI: 10.18653/v1/D19-1498. URL: https://www.aclweb.org/anthology/D19-1498.

Collins-Thompson, K. and J. Callan. (2005). "Query expansion using random walk models". In: *Proceedings of the 14th ACM international conference on Information and knowledge management*. 704–711.

Cui, L., Y. Wu, S. Liu, Y. Zhang, and M. Zhou. (2020a). "MuTual: A Dataset for Multi-Turn Dialogue Reasoning". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 1406–1416.

Cui, P., L. Hu, and Y. Liu. (2020b). "Enhancing Extractive Text Summarization with Topic-Aware Graph Neural Networks". *arXiv preprint arXiv:2010.06253*.

Cui, S., B. Yu, T. Liu, Z. Zhang, X. Wang, and J. Shi. (2020c). "Edge-Enhanced Graph Convolution Networks for Event Detection with Syntactic Relation". In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics. 2329–2339. DOI: 10.18653/v1/2020.findings-emnlp.211. URL: https://www.aclweb.org/anthology/2020.findings-emnlp.211.

Cui, Y., Z. Chen, S. Wei, S. Wang, T. Liu, and G. Hu. (2017). "Attention-over-Attention Neural Networks for Reading Comprehension". In: *ACL (1)*.

Dahl, D. A., M. Bates, M. Brown, W. Fisher, K. Hunicke-Smith, D. Pallett, C. Pao, A. Rudnicky, and E. Shriberg. (1994). "Expanding the Scope of the ATIS Task: The ATIS-3 Corpus". In: *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*. URL: https://www.aclweb.org/anthology/H94-1010.

Damonte, M. and S. B. Cohen. (2019). "Structural Neural Encoders for AMR-to-text Generation". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics. 3649–3658. DOI: 10.18653/v1/N19-1366. URL: https://www.aclweb.org/anthology/N19-1366.

De Cao, N., W. Aziz, and I. Titov. (2018). "Question answering by reasoning across documents with graph convolutional networks".

Defferrard, M., X. Bresson, and P. Vandergheynst. (2016). "Convolutional neural networks on graphs with fast localized spectral filtering". *Advances in neural information processing systems*. 29.

Dettmers, T., P. Minervini, P. Stenetorp, and S. Riedel. (2018a). "Convolutional 2d knowledge graph embeddings". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. No. 1.

Dettmers, T., M. Pasquale, S. Pontus, and S. Riedel. (2018b). "Convolutional 2D Knowledge Graph Embeddings". In: *Proceedings of the 32th AAAI Conference on Artificial Intelligence*. 1811–1818. URL: https://arxiv.org/abs/1707.01476.

Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova. (2018). "Bert: Pre-training of deep bidirectional transformers for language understanding". *arXiv preprint arXiv:1810.04805*.

Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova. (2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://www.aclweb.org/anthology/N19-1423.

Ding, M., C. Zhou, Q. Chen, H. Yang, and J. Tang. (2019a). "Cognitive Graph for Multi-Hop Reading Comprehension at Scale". In: *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. Ed. by A. Korhonen, D. R. Traum, and L. Màrquez. Association for Computational Linguistics. 2694–2703.

Ding, R., P. Xie, X. Zhang, W. Lu, L. Li, and L. Si. (2019b). "A Neural Multi-digraph Model for Chinese NER with Gazetteers". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics. 1462–1467. DOI: 10.18653/v1/P19-1141. URL: https://www.aclweb.org/anthology/P19-1141.

Do, B.-N. and I. Rehbein. (2020). "Neural Reranking for Dependency Parsing: An Evaluation". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics. 4123–4133. DOI: 10.18653/v1/2020.acl-main.379. URL: https://www.aclweb.org/anthology/2020.acl-main.379.

Do, K., T. Tran, and S. Venkatesh. (2019). "Graph transformation policy network for chemical reaction prediction". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 750–760.

Dong, L. and M. Lapata. (2016). "Language to Logical Form with Neural Attention". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics. 33–43. DOI: 10.18653/v1/P16-1004. URL: https://www.aclweb.org/anthology/P16-1004.

Dong, L., F. Wei, C. Tan, D. Tang, M. Zhou, and K. Xu. (2014). "Adaptive recursive neural network for target-dependent twitter sentiment classification". In: *Proceedings of the 52nd annual meeting of the association for computational linguistics (volume 2: Short papers)*. 49–54.

Dozat, T. and C. D. Manning. (2016). "Deep biaffine attention for neural dependency parsing". *arXiv preprint arXiv:1611.01734*.

Du, X., J. Shao, and C. Cardie. (2017). "Learning to Ask: Neural Question Generation for Reading Comprehension". In: *ACL (1)*.

Dua, D., Y. Wang, P. Dasigi, G. Stanovsky, S. Singh, and M. Gardner. (2019). "DROP: A Reading Comprehension Benchmark Requiring Discrete Reasoning Over Paragraphs". In: *Proceedings of NAACL-HLT*. 2368–2378.

Dyer, C., A. Kuncoro, M. Ballesteros, and N. A. Smith. (2016). "Recurrent Neural Network Grammars". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics. 199–209. DOI: 10.18653/v1/N16-1024. URL: https://www.aclweb.org/anthology/N16-1024.

Edouard, A., E. Cabrio, S. Tonelli, and N. Le-Thanh. (2017). "Graph-based Event Extraction from Twitter". In: *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*. Varna, Bulgaria: INCOMA Ltd. 222–230. DOI: 10.26615/978-954-452-049-6_031. URL: https://doi.org/10.26615/978-954-452-049-6_031.

Elliott, D., S. Frank, K. Sima'an, and L. Specia. (2016). "Multi30K: Multilingual English-German Image Descriptions". In: *Proceedings of the 5th Workshop on Vision and Language*. Berlin, Germany: Association for Computational Linguistics. 70–74. DOI: 10.18653/v1/W16-3210. URL: https://www.aclweb.org/anthology/W16-3210.

Eric, M., R. Goel, S. Paul, A. Sethi, S. Agarwal, S. Gao, A. Kumar, A. K. Goyal, P. Ku, and D. Hakkani-Tür. (2020). "MultiWOZ 2.1: A Consolidated Multi-Domain Dialogue Dataset with State Corrections and State Tracking Baselines". In: *LREC*.

Eriguchi, A., K. Hashimoto, and Y. Tsuruoka. (2016). "Tree-to-Sequence Attentional Neural Machine Translation". In: *ACL (1)*.

Erkan, G. (2006). "Language model-based document clustering using random walks". In: *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*. 479–486.

Fabbri, A. R., I. Li, T. She, S. Li, and D. R. Radev. (2019). "Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model". *arXiv preprint arXiv:1906.01749*.

Fan, S., J. Zhu, X. Han, C. Shi, L. Hu, B. Ma, and Y. Li. (2019). "Metapath-guided heterogeneous graph neural network for intent recommendation". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2478–2486.

Fancellu, F., S. Gilroy, A. Lopez, and M. Lapata. (2019). "Semantic graph parsing with recurrent neural network DAG grammars". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2769–2778.

Fang, Y., S. Sun, Z. Gan, R. Pillai, S. Wang, and J. Liu. (2020a). "Hierarchical Graph Network for Multi-hop Question Answering". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 8823–8838.

Fang, Y., S. Sun, Z. Gan, R. Pillai, S. Wang, and J. Liu. (2020b). "Hierarchical Graph Network for Multi-hop Question Answering". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*. Ed. by B. Webber, T. Cohn, Y. He, and Y. Liu. Association for Computational Linguistics. 8823–8838.

Fei, H., M. Zhang, F. Li, and D. Ji. (2020). "Cross-lingual semantic role labeling with model transfer". *IEEE/ACM Transactions on Audio, Speech, and Language Processing*. 28: 2427–2437.

Feng, Y., X. Chen, B. Y. Lin, P. Wang, J. Yan, and X. Ren. (2020a). "Scalable Multi-Hop Relational Reasoning for Knowledge-Aware Question Answering". *arXiv preprint arXiv:2005.00646*.

Feng, Y., X. Chen, B. Y. Lin, P. Wang, J. Yan, and X. Ren. (2020b). "Scalable Multi-Hop Relational Reasoning for Knowledge-Aware Question Answering". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*. Ed. by B. Webber, T. Cohn, Y. He, and Y. Liu. Association for Computational Linguistics. 1295–1309.

Fernandes, P., M. Allamanis, and M. Brockschmidt. (2019). "Structured Neural Summarization". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=H1ersoRqtm.

Ferreira, D. and A. Freitas. (2020). "Premise Selection in Natural Language Mathematical Texts". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics. 7365–7374. DOI: 10.18653/v1/2020.acl-main.657. URL: https://www.aclweb.org/anthology/2020.acl-main.657.

Fey, M. and J. E. Lenssen. (2019). "Fast graph representation learning with PyTorch Geometric". *arXiv preprint arXiv:1903.02428*.

Flanigan, J., S. Thomson, J. G. Carbonell, C. Dyer, and N. A. Smith. (2014). "A discriminative graph-based parser for the abstract meaning representation". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1426–1436.

Franceschi, L., M. Niepert, M. Pontil, and X. He. "Learning Discrete Structures for Graph Neural Networks". In: *Proceedings of the 36th International Conference on Machine, volume = 97, pages = 1972–1982, year = 2019*.

Fu, Q., L. Song, W. Du, and Y. Zhang. (2021). "End-to-End AMR Coreference Resolution". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 4204–4214.

Fu, T.-J., P.-H. Li, and W.-Y. Ma. (2019). "GraphRel: Modeling text as relational graphs for joint entity and relation extraction". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 1409–1418.

Gao, H., L. Wu, P. Hu, and F. Xu. (2020). "RDF-to-Text Generation with Graph-augmented Structural Neural Encoders". In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*. International Joint Conferences on Artificial Intelligence Organization. 3030–3036.

Gao, H., Y. Chen, and S. Ji. (2019). "Learning graph pooling and hybrid convolutional operations for text representations". In: *The World Wide Web Conference*. 2743–2749.

Gao, Q. and S. Vogel. (2011). "Corpus expansion for statistical machine translation with semantic role label substitution rules". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. 294–298.

Gardent, C., A. Shimorina, S. Narayan, and L. Perez-Beltrachini. (2017). "Creating Training Corpora for NLG Micro-Planners". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics. 179–188. DOI: 10.18653/v1/P17-1017. URL: http://www.aclweb.org/anthology/P17-1017.

Gehring, J., M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. (2017). "Convolutional sequence to sequence learning". In: *International Conference on Machine Learning*. PMLR. 1243–1252.

Gehrmann, S., Y. Deng, and A. M. Rush. (2018). "Bottom-up abstractive summarization". *arXiv preprint arXiv:1808.10792*.

Ghosal, D., D. Hazarika, A. Roy, N. Majumder, R. Mihalcea, and S. Poria. (2020). "KinGDOM: Knowledge-Guided DOMain Adaptation for Sentiment Analysis". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics. 3198–3210. DOI: 10.18653/v1/2020.acl-main.292. URL: https://www.aclweb.org/anthology/2020.acl-main.292.

Gilmer, J., S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. (2017). "Neural message passing for quantum chemistry". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 1263–1272.

Goldberg, A. B. and X. Zhu. (2006). "Seeing stars when there aren't many stars: Graph-based semi-supervised learning for sentiment categorization". In: *Proceedings of TextGraphs: The first workshop on graph based methods for natural language processing*. 45–52.

Gómez-Bombarelli, R., J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik. (2018). "Automatic chemical design using a data-driven continuous representation of molecules". *ACS central science*. 4(2): 268–276.

Gu, J., Z. Lu, H. Li, and V. O. Li. (2016). "Incorporating copying mechanism in sequence-to-sequence learning". *arXiv preprint arXiv:1603.06393*.

Gui, T., Y. Zou, Q. Zhang, M. Peng, J. Fu, Z. Wei, and X. Huang. (2019). "A Lexicon-Based Graph Neural Network for Chinese NER". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics. 1040–1050. DOI: 10.18653/v1/D19-1096. URL: https://www.aclweb.org/anthology/D19-1096.

Guo, X., L. Wu, and L. Zhao. (2018). "Deep graph translation". *arXiv preprint arXiv:1805.09980*.

Guo, X., L. Zhao, C. Nowzari, S. Rafatirad, H. Homayoun, and S. M. P. Dinakarrao. (2019a). "Deep multi-attributed graph translation with node-Edge Co-evolution". In: *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE. 250–259.

Guo, Z., Y. Zhang, and W. Lu. (2019b). "Attention Guided Graph Convolutional Networks for Relation Extraction". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics. 241–251. DOI: 10.18653/v1/P19-1024. URL: https://www.aclweb.org/anthology/P19-1024.

Guo, Z., Y. Zhang, Z. Teng, and W. Lu. (2019c). "Densely Connected Graph Convolutional Networks for Graph-to-Sequence Learning". *Transactions of the Association for Computational Linguistics*. 7(Mar.): 297–312. DOI: 10.1162/tacl_a_00269. URL: https://www.aclweb.org/anthology/Q19-1019.

Gupta, S., S. Kenkre, and P. Talukdar. (2019). "CaRe: Open Knowledge Graph Embeddings". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics. 378–388. DOI: 10.18653/v1/D19-1036. URL: https://www.aclweb.org/anthology/D19-1036.

Haghighi, A., A. Y. Ng, and C. D. Manning. (2005). "Robust textual inference via graph matching". In: *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. 387–394.

Hamilton, W., Z. Ying, and J. Leskovec. (2017a). "Inductive representation learning on large graphs". In: *Advances in Neural Information Processing Systems*. 1024–1034.

Hamilton, W. L., R. Ying, and J. Leskovec. (2017b). "Representation learning on graphs: Methods and applications". *arXiv preprint arXiv:1709.05584*.

Han, J., B. Cheng, and X. Wang. (2020). "Open Domain Question Answering based on Text Enhanced Knowledge Graph with Hyperedge Infusion". In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics. 1475–1481. DOI: 10.18653/v1/2020.findings-emnlp.133. URL: https://www.aclweb.org/anthology/2020.findings-emnlp.133.

Hashimoto, K. and Y. Tsuruoka. (2017). "Neural Machine Translation with Source-Side Latent Graph Parsing". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics. 125–135. DOI: 10.18653/v1/D17-1012. URL: https://www.aclweb.org/anthology/D17-1012.

Haveliwala, T. H. (2002). "Topic-Sensitive PageRank". In: *Proceedings of the 11th International Conference on World Wide Web. WWW '02*. Honolulu, Hawaii, USA: Association for Computing Machinery. 517–526. ISBN: 1581134495. DOI: 10.1145/511446.511513. URL: https://doi.org/10.1145/511446.511513.

He, B., D. Zhou, J. Xiao, X. Jiang, Q. Liu, N. J. Yuan, and T. Xu. (2020). "Integrating Graph Contextualized Knowledge into Pre-trained Language Models". In: *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*. Vol. EMNLP 2020. *Findings of ACL*. Association for Computational Linguistics. 2281–2290.

He, L., K. Lee, O. Levy, and L. Zettlemoyer. (2018). "Jointly predicting predicates and arguments in neural semantic role labeling". *arXiv preprint arXiv:1805.04787*.

He, L., K. Lee, M. Lewis, and L. Zettlemoyer. (2017). "Deep semantic role labeling: What works and what's next". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 473–483.

Henaff, M., J. Bruna, and Y. LeCun. (2015). "Deep convolutional networks on graph-structured data". *arXiv preprint arXiv:1506.05163*.

Hermann, K. M., T. Kociskỳ, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. (2015). "Teaching Machines to Read and Comprehend". In: *NIPS*.

Hersh, W., C. Buckley, T. Leone, and D. Hickam. (1994). "OHSUMED: An interactive retrieval evaluation and new large test collection for research". In: *SIGIR'94*. Springer. 192–201.

Hochreiter, S. and J. Schmidhuber. (1997). "Long short-term memory". *Neural computation*. 9(8): 1735–1780.

Hoffman, M. D., D. M. Blei, C. Wang, and J. Paisley. (2013). "Stochastic variational inference". *The Journal of Machine Learning Research*. 14(1): 1303–1347.

Hofmann, T. (1999). "Probabilistic latent semantic indexing". In: *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. 50–57.

Hu, B., Z. Lu, H. Li, and Q. Chen. (2014). "Convolutional Neural Network Architectures for Matching Natural Language Sentences". In: *NIPS*.

Hu, J., Q. Fang, S. Qian, and C. Xu. (2020a). "Multi-Modal Attentive Graph Pooling Model for Community Question Answer Matching". In: *Proceedings of the 28th ACM International Conference on Multimedia. MM '20*. Seattle, WA, USA: Association for Computing Machinery. 3505–3513. ISBN: 9781450379885. DOI: 10.1145/3394171.3413711. URL: https://doi.org/10.1145/3394171.3413711.

Hu, J., Q. Fang, S. Qian, and C. Xu. (2020b). "Multi-modal Attentive Graph Pooling Model for Community Question Answer Matching". In: *Proceedings of the 28th ACM International Conference on Multimedia*. 3505–3513.

Hu, J., S. Qian, Q. Fang, and C. Xu. (2018). "Attentive Interactive Convolutional Matching for Community Question Answering in Social Multimedia". In: *Proceedings of the 26th ACM International Conference on Multimedia*. *MM '18*. Seoul, Republic of Korea: Association for Computing Machinery. 456–464. ISBN: 9781450356657. DOI: 10.1145/3240508.3240626. URL: https://doi.org/10.1145/3240508.3240626.

Hu, J., S. Qian, Q. Fang, and C. Xu. (2019a). "Hierarchical Graph Semantic Pooling Network for Multi-Modal Community Question Answer Matching". In: *Proceedings of the 27th ACM International Conference on Multimedia*. *MM '19*. Nice, France: Association for Computing Machinery. 1157–1165. ISBN: 9781450368896. DOI: 10.1145/3343031.3350966. URL: https://doi.org/10.1145/3343031.3350966.

Hu, J., S. Qian, Q. Fang, and C. Xu. (2019b). "Hierarchical graph semantic pooling network for multi-modal community question answer matching". In: *Proceedings of the 27th ACM International Conference on Multimedia*. 1157–1165.

Hu, L., T. Yang, C. Shi, H. Ji, and X. Li. (2019c). "Heterogeneous Graph Attention Networks for Semi-supervised Short Text Classification". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*. Ed. by K. Inui, J. Jiang, V. Ng, and X. Wan. Association for Computational Linguistics. 4820–4829.

Hu, W., B. Liu, J. Gomes, M. Zitnik, P. Liang, V. Pande, and J. Leskovec. (2019d). "Strategies for pre-training graph neural networks". *arXiv preprint arXiv:1905.12265*.

Hu, W., Z. Chan, B. Liu, D. Zhao, J. Ma, and R. Yan. (2019e). "GSN: A Graph-Structured Network for Multi-Party Dialogues". In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*. Ed. by S. Kraus. ijcai.org. 5010–5016.

Hu, Z., Y. Dong, K. Wang, and Y. Sun. (2020c). "Heterogeneous graph transformer". In: *Proceedings of The Web Conference 2020*. 2704–2710.

Huang, B. and K. Carley. (2018). "Parameterized Convolutional Neural Networks for Aspect Level Sentiment Classification". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics. 1091–1096. DOI: 10.18653/v1/D18-1136. URL: https://www.aclweb.org/anthology/D18-1136.

Huang, B. and K. Carley. (2019). "Syntax-Aware Aspect Level Sentiment Classification with Graph Attention Networks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics. 5469–5477. DOI: 10.18653/v1/D19-1549. URL: https://www.aclweb.org/anthology/D19-1549 (accessed on 12/26/2020).

Huang, D., P. Chen, R. Zeng, Q. Du, M. Tan, and C. Gan. (2020a). "Location-Aware Graph Convolutional Networks for Video Question Answering". In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence*. AAAI Press. 11021–11028.

Huang, L., D. Ma, S. Li, X. Zhang, and H. Wang. (2019). "Text Level Graph Neural Network for Text Classification". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*. Ed. by K. Inui, J. Jiang, V. Ng, and X. Wan. Association for Computational Linguistics. 3442–3448.

Huang, L., L. Wu, and L. Wang. (2020b). "Knowledge Graph-Augmented Abstractive Summarization with Semantic-Driven Cloze Reward". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics. 5094–5107. DOI: 10.18653/v1/2020.acl-main.457. URL: https://www.aclweb.org/anthology/2020.acl-main.457.

Hughes, T. and D. Ramage. (2007). "Lexical semantic relatedness with random graph walks". In: *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*. 581–589.

Huo, S., T. Ma, J. Chen, M. Chang, L. Wu, and M. Witbrock. (2019). "Graph Enhanced Cross-Domain Text-to-SQL Generation". In: *Proceedings of the Thirteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-13)*. Hong Kong: Association for Computational Linguistics. 159–163. DOI: 10.18653/v1/D19-5319. URL: https://www.aclweb.org/anthology/D19-5319.

Isonuma, M., J. Mori, D. Bollegala, and I. Sakata. (2020). "Tree-Structured Neural Topic Model". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics. 800–806. DOI: 10.18653/v1/2020.acl-main.73. URL: https://www.aclweb.org/anthology/2020.acl-main.73.

Iyer, S., I. Konstas, A. Cheung, and L. Zettlemoyer. (2016). "Summarizing source code using a neural attention model". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2073–2083.

Ji, S., S. Pan, E. Cambria, P. Marttinen, and P. S. Yu. (2020). "A survey on knowledge graphs: Representation, acquisition and applications". *arXiv preprint arXiv:2002.00388*.

Ji, T., Y. Wu, and M. Lan. (2019). "Graph-based Dependency Parsing with Graph Neural Networks". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics. 2475–2485. DOI: 10.18653/v1/P19-1237. URL: https://www.aclweb.org/anthology/P19-1237.

Jia, R. and P. Liang. (2016). "Data Recombination for Neural Semantic Parsing". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics. 12–22. DOI: 10.18653/v1/P16-1002. URL: https://www.aclweb.org/anthology/P16-1002.

Jia, R., Y. Cao, H. Tang, F. Fang, C. Cao, and S. Wang. (2020). "Neural Extractive Summarization with Hierarchical Attentive Heterogeneous Graph Network". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics. 3622–3631. DOI: 10.18653/v1/2020.emnlp-main.295. URL: https://www.aclweb.org/anthology/2020.emnlp-main.295.

Jiang, L., M. Yu, M. Zhou, X. Liu, and T. Zhao. (2011). "Target-dependent twitter sentiment classification". In: *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*. 151–160.

Jiang, Q., L. Chen, R. Xu, X. Ao, and M. Yang. (2019). "A challenge dataset and effective models for aspect-based sentiment analysis". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 6281–6286.

Jin, H., L. Hou, J. Li, and T. Dong. (2019). "Fine-Grained Entity Typing via Hierarchical Multi Graph Convolutional Networks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics. 4969–4978. DOI: 10.18653/v1/D19-1502. URL: https://www.aclweb.org/anthology/D19-1502.

Jin, H., T. Wang, and X. Wan. (2020a). "SemSUM: Semantic Dependency Guided Neural Abstractive Summarization". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. No. 05. 8026–8033.

Jin, H., T. Wang, and X. Wan. (2020b). "SemSUM: Semantic Dependency Guided Neural Abstractive Summarization". *Proceedings of the AAAI Conference on Artificial Intelligence*. 34(05): 8026–8033. DOI: 10.1609/aaai.v34i05.6312. URL: https://ojs.aaai.org/index.php/AAAI/article/view/6312.

Jin, L. and D. Gildea. (2020). "Generalized Shortest-Paths Encoders for AMR-to-Text Generation". In: *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics. 2004–2013. URL: https://www.aclweb.org/anthology/2020.coling-main.181.

Joulin, A., É. Grave, P. Bojanowski, and T. Mikolov. (2017). "Bag of Tricks for Efficient Text Classification". In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. 427–431.

Kalofolias, V. and N. Perraudin. (2019). "Large Scale Graph Learning From Smooth Signals". In: *7th International Conference on Learning Representations*.

Kapanipathi, P., V. Thost, S. Patel, S. Whitehead, I. Abdelaziz, A. Balakrishnan, M. Chang, K. P. Fadnis, R. C. Gunasekara, B. Makni, N. Mattei, K. Talamadupula, and A. Fokoue. (2020). "Infusing Knowledge into the Textual Entailment Task Using Graph Convolutional Networks". In: *AAAI*.

Katharopoulos, A., A. Vyas, N. Pappas, and F. Fleuret. (2020). "Transformers are rnns: Fast autoregressive transformers with linear attention". In: *International Conference on Machine Learning*. PMLR. 5156–5165.

Khot, T., A. Sabharwal, and P. Clark. (2018). "Scitail: A textual entailment dataset from science question answering". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. No. 1.

Kim, Y. (2014). "Convolutional Neural Networks for Sentence Classification". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. Ed. by A. Moschitti, B. Pang, and W. Daelemans. ACL. 1746–1751. DOI: 10.3115/v1/d14-1181. URL: https://doi.org/10.3115/v1/d14-1181.

Kingsbury, P. R. and M. Palmer. (2002). "From TreeBank to PropBank." In: *LREC*. Citeseer. 1989–1993.

Kiperwasser, E. and Y. Goldberg. (2016). "Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations". *Transactions of the Association for Computational Linguistics*. 4: 313–327. DOI: 10.1162/tacl_a_00101. URL: https://www.aclweb.org/anthology/Q16-1023.

Kipf, T. N. and M. Welling. (2016). "Semi-supervised classification with graph convolutional networks". *arXiv preprint arXiv:1609.02907*.

Kiros, R., Y. Zhu, R. Salakhutdinov, R. S. Zemel, R. Urtasun, A. Torralba, and S. Fidler. (2015). "Skip-Thought Vectors". In: *NIPS*.

Klicpera, J., A. Bojchevski, and S. Günnemann. (2019). "Combining Neural Networks with Personalized PageRank for Classification on Graphs". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=H1gL-2A9Ym.

Knight, K., B. Badarau, L. Banarescu, C. Bonial, M. Bardocz, K. Griffitt, U. Hermjakob, D. Marcu, M. Palmer, T. O'Gorman, *et al.* (2017). "Abstract Meaning Representation (AMR) Annotation Release 2.0". *Tech. rep.* Technical Report LDC2017T10, Linguistic Data Consortium, Philadelphia, PA, June.

Knight, K., L. Baranescu, C. Bonial, M. Georgescu, K. Griffitt, U. Hermjakob, D. Marcu, M. Palmer, and N. Schneider. (2014). "Abstract meaning representation (AMR) annotation release 1.0 LDC2014T12". *Web Download. Philadelphia: Linguistic Data Consortium*.

Koncel-Kedziorski, R., D. Bekal, Y. Luan, M. Lapata, and H. Hajishirzi. (2019). "Text Generation from Knowledge Graphs with Graph Transformers". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics. 2284–2293. DOI: 10.18653/v1/N19-1238. URL: https://www.aclweb.org/anthology/N19-1238.

Koncel-Kedziorski, R., S. Roy, A. Amini, N. Kushman, and H. Hajishirzi. (2016). "MAWPS: A math word problem repository". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 1152–1157.

Krizhevsky, A., I. Sutskever, and G. E. Hinton. (2012). "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger. Vol. 25. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

Kumar, V., Y. Hua, G. Ramakrishnan, G. Qi, L. Gao, and Y.-F. Li. (2019). "Difficulty-controllable multi-hop question generation from knowledge graphs". In: *International Semantic Web Conference*. Springer. 382–398.

Lafferty, J. D., A. McCallum, and F. C. N. Pereira. (2001). "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data". In: *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*. Ed. by C. E. Brodley and A. P. Danyluk. Morgan Kaufmann. 282–289.

Lan, Z., M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. (2019). "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations". In: *International Conference on Learning Representations*.

Lang, K. (1995). "Newsweeder: Learning to filter netnews". In: *Machine Learning Proceedings 1995*. Elsevier. 331–339.

Larochelle, H. and S. Lauly. (2012). "A neural autoregressive topic model". *Advances in Neural Information Processing Systems*. 25: 2708–2716.

Le, Q. and T. Mikolov. (2014). "Distributed representations of sentences and documents". In: *International conference on machine learning*. PMLR. 1188–1196.

LeClair, A., S. Haque, L. Wu, and C. McMillan. (2020). "Improved code summarization via a graph neural network". *arXiv preprint arXiv:2004.02843*.

LeCun, Y. and Y. Bengio. (1998). "Convolutional networks for images, speech, and time series". In: *The handbook of brain theory and neural networks*. 255–258.

Lee, D., C. Szegedy, M. Rabe, S. Loos, and K. Bansal. (2020). "Mathematical Reasoning in Latent Space". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=Ske31kBtPr.

Lee, H., Y. Peirsman, A. Chang, N. Chambers, M. Surdeanu, and D. Jurafsky. (2011). "Stanford's multi-pass sieve coreference resolution system at the conll-2011 shared task". In: *Proceedings of the 15th conference on computational natural language learning: Shared task*. Association for Computational Linguistics. 28–34.

Levesque, H., E. Davis, and L. Morgenstern. (2012). "The winograd schema challenge". In: *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*. Citeseer.

Levi, F. W. (1942). *Finite geometrical systems: six public lectues delivered in February, 1940, at the University of Calcutta*. University of Calcutta.

Lewis, D. D., Y. Yang, T. G. Rose, and F. Li. (2004). "Rcv1: A new benchmark collection for text categorization research". *Journal of machine learning research*. 5(Apr): 361–397.

Li, C. and D. Goldwasser. (2019). "Encoding Social Information with Graph Convolutional Networks forPolitical Perspective Detection in News Media". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics. 2594–2604. DOI: 10.18653/v1/P19-1247. URL: https://www.aclweb.org/anthology/P19-1247.

Li, C., Y. Cao, L. Hou, J. Shi, J. Li, and T.-S. Chua. (2019). "Semi-supervised entity alignment via joint knowledge embedding model and cross-graph model". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2723–2732.

Li, L., A. Way, and Q. Liu. (2017). "Context-Aware Graph Segmentation for Graph-Based Translation". In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Valencia, Spain: Association for Computational Linguistics. 599–604. URL: https://www.aclweb.org/anthology/E17-2095.

Li, R., S. Wang, F. Zhu, and J. Huang. (2018a). "Adaptive graph convolutional neural networks". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32.

Li, S., L. Wu, S. Feng, F. Xu, F. Xu, and S. Zhong. (2020a). "Graph-to-Tree Neural Networks for Learning Structured Input-Output Translation with Applications to Semantic Parsing and Math Word Problem". In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics. 2841–2852. DOI: 10.18653/v1/2020.findings-emnlp.255. URL: https://www.aclweb.org/anthology/2020.findings-emnlp.255.

Li, W., X. Xiao, J. Liu, H. Wu, H. Wang, and J. Du. (2020b). "Leveraging Graph to Improve Abstractive Multi-Document Summarization". *arXiv preprint arXiv:2005.10043*.

Li, Y., J. Amelot, X. Zhou, S. Bengio, and S. Si. (2020c). "Auto completion of user interface layout design using transformer-based tree decoders". *arXiv preprint arXiv:2001.05308*.

Li, Y., N. Duan, B. Zhou, X. Chu, W. Ouyang, X. Wang, and M. Zhou. (2018b). "Visual question generation as dual task of visual question answering". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 6116–6124.

Li, Y., D. Tarlow, M. Brockschmidt, and R. Zemel. (2015). "Gated graph sequence neural networks". *arXiv preprint arXiv:1511.05493*.

Li, Z., S. He, J. Cai, Z. Zhang, H. Zhao, G. Liu, L. Li, and L. Si. (2018c). "A unified syntax-aware framework for semantic role labeling". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2401–2411.

Liao, K., L. Lebanoff, and F. Liu. (2018). "Abstract meaning representation for multi-document summarization". *arXiv preprint arXiv:1806.05655*.

Lin, B. Y., X. Chen, J. Chen, and X. Ren. (2019a). "KagNet: Knowledge-Aware Graph Networks for Commonsense Reasoning". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics. 2829–2839. DOI: 10.18653/v1/D19-1282. URL: https://www.aclweb.org/anthology/D19-1282.

Lin, B. Y., X. Chen, J. Chen, and X. Ren. (2019b). "KagNet: Knowledge-Aware Graph Networks for Commonsense Reasoning". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2822–2832.

Lin, X. V., R. Socher, and C. Xiong. (2018). "Multi-Hop Knowledge Graph Reasoning with Reward Shaping". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 3243–3253.

Linmei, H., T. Yang, C. Shi, H. Ji, and X. Li. (2019). "Heterogeneous Graph Attention Networks for Semi-supervised Short Text Classification". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics. 4821–4830. DOI: 10.18653/v1/D19-1488. URL: https://www.aclweb.org/anthology/D19-1488.

Liu, B., D. Niu, H. Wei, J. Lin, Y. He, K. Lai, and Y. Xu. (2019a). "Matching Article Pairs with Graphical Decomposition and Convolutions". In: *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. Ed. by A. Korhonen, D. R. Traum, and L. Màrquez. Association for Computational Linguistics. 6284–6294.

Liu, B. and L. Wu. (2022). "Graph Neural Networks in Natural Language Processing". In: *Graph Neural Networks: Foundations, Frontiers, and Applications*. Ed. by L. Wu, P. Cui, J. Pei, and L. Zhao. Singapore: Springer Singapore. 463–481.

Liu, B., M. Zhao, D. Niu, K. Lai, Y. He, H. Wei, and Y. Xu. (2019b). "Learning to Generate Questions by LearningWhat Not to Generate". In: *The World Wide Web Conference*. *WWW '19*. San Francisco, CA, USA: Association for Computing Machinery. 1106–1118.

Liu, B., B. Ramsundar, P. Kawthekar, J. Shi, J. Gomes, Q. Luu Nguyen, S. Ho, J. Sloane, P. Wender, and V. Pande. (2017). "Retrosynthetic reaction prediction using neural sequence-to-sequence models". *ACS central science*. 3(10): 1103–1113.

Liu, D. and D. Gildea. (2010). "Semantic role features for machine translation". In: *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*. 716–724.

Liu, J. and Y. Zhang. (2017). "In-Order Transition-based Constituent Parsing". *Transactions of the Association for Computational Linguistics*. 5: 413–424. DOI: 10.1162/tacl_a_00070. URL: https://www.aclweb.org/anthology/Q17-1029.

Liu, M., Y. Luo, L. Wang, Y. Xie, H. Yuan, S. Gui, H. Yu, Z. Xu, J. Zhang, Y. Liu, *et al.* (2021a). "DIG: A Turnkey Library for Diving into Graph Deep Learning Research". *arXiv preprint arXiv:2103.12608*.

Liu, P., S. Chang, X. Huang, J. Tang, and J. C. K. Cheung. (2019c). "Contextualized non-local neural networks for sequence learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 6762–6769.

Liu, P., X. Qiu, and X. Huang. (2016). "Recurrent neural network for text classification with multi-task learning". In: *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. 2873–2879.

Liu, P. J., M. Saleh, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser, and N. Shazeer. (2018a). "Generating wikipedia by summarizing long sequences". *arXiv preprint arXiv:1801.10198*.

Liu, S., Y. Chen, X. Xie, J. K. Siow, and Y. Liu. (2021b). "Retrieval-Augmented Generation for Code Summarization via Hybrid GNN". In: *9th International Conference on Learning Representations*.

Liu, X., Z. Luo, and H. Huang. (2018b). "Jointly Multiple Events Extraction via Attention-based Graph Information Aggregation". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics. 1247–1256. DOI: 10.18653/v1/D18-1156. URL: https://www.aclweb.org/anthology/D18-1156.

Liu, X., X. You, X. Zhang, J. Wu, and P. Lv. (2020). "Tensor Graph Convolutional Networks for Text Classification". In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press. 8409–8416.

Liu, X., Q. Chen, C. Deng, H. Zeng, J. Chen, D. Li, and B. Tang. (2018c). "Lcqmc: A large-scale chinese question matching corpus". In: *Proceedings of the 27th International Conference on Computational Linguistics*. 1952–1962.

Liu, Y. (2019). "Fine-tune BERT for extractive summarization". *arXiv preprint arXiv:1903.10318*.

Liu, Y., M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. (2019d). "Roberta: A robustly optimized bert pretraining approach". *arXiv preprint arXiv:1907.11692*.

Liu, Y., S. Feng, D. Wang, K. Song, F. Ren, and Y. Zhang. (2021c). "A Graph Reasoning Network for Multi-turn Response Selection via Customized Pretraining". In: *The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Virtual, February 2-9, 2021*.

Long, Q., Y. Jin, G. Song, Y. Li, and W. Lin. (2020). "Graph Structural-topic Neural Network". In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. *KDD '20*. New York, NY, USA: Association for Computing Machinery. 1065–1073. ISBN: 978-1-4503-7998-4. DOI: 10.1145/3394486.3403150. URL: http://doi.org/10.1145/3394486.3403150 (accessed on 12/25/2020).

Lowe, R., N. Pow, I. V. Serban, and J. Pineau. (2015). "The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems". In: *16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. 285.

Lu, Z. and H. Li. (2013). "A deep architecture for matching short texts". *Advances in neural information processing systems*. 26: 1367–1375.

Luan, Y., L. He, M. Ostendorf, and H. Hajishirzi. (2018). "Multi-Task Identification of Entities, Relations, and Coreference for Scientific Knowledge Graph Construction". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 3219–3232.

Luan, Y., D. Wadden, L. He, A. Shah, M. Ostendorf, and H. Hajishirzi. (2019). "A general framework for information extraction using dynamic span graphs". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 3036–3046.

Luke, S. (2005). "Ze lemoyer and Michael Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars". In: *UAI*. Vol. 2. 3.

Luo, Y. and H. Zhao. (2020). "Bipartite Flat-Graph Network for Nested Named Entity Recognition". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics. 6408–6418. DOI: 10.18653/v1/2020.acl-main.571. URL: https://www.aclweb.org/anthology/2020.acl-main.571.

Luong, M.-T., H. Pham, and C. D. Manning. (2015). "Effective approaches to attention-based neural machine translation". *arXiv preprint arXiv:1508.04025*.

Ma, X., Z. Hu, J. Liu, N. Peng, G. Neubig, and E. Hovy. (2018). "Stack-Pointer Networks for Dependency Parsing". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics. 1403–1414. DOI: 10.18653/v1/P18-1130. URL: https://www.aclweb.org/anthology/P18-1130.

Ma, Y., S. Wang, C. C. Aggarwal, and J. Tang. (2019). "Graph convolutional networks with eigenpooling". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 723–731.

Malaviya, C., C. Bhagavatula, A. Bosselut, and Y. Choi. (2020). "Commonsense Knowledge Base Completion with Structural and Semantic Context." In: *AAAI*. 2925–2933.

Mann, W. C. and S. A. Thompson. (1987). *Rhetorical structure theory: A theory of text organization*. University of Southern California, Information Sciences Institute Los Angeles.

Marcheggiani, D., J. Bastings, and I. Titov. (2018). "Exploiting Semantics in Neural Machine Translation with Graph Convolutional Networks". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. New Orleans, Louisiana: Association for Computational Linguistics. 486–492. DOI: 10.18653/v1/N18-2078. URL: https://www.aclweb.org/anthology/N18-2078.

Marcheggiani, D., A. Frolov, and I. Titov. (2017). "A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling". *arXiv preprint arXiv:1701.02593*.

Marcheggiani, D. and L. Perez-Beltrachini. (2018). "Deep Graph Convolutional Encoders for Structured Data to Text Generation". In: *Proceedings of the 11th International Conference on Natural Language Generation*. Tilburg University, The Netherlands: Association for Computational Linguistics. 1–9. DOI: 10.18653/v1/W18-6501. URL: https://www.aclweb.org/anthology/W18-6501.

Marcheggiani, D. and I. Titov. (2017). "Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 1506–1515.

Marcheggiani, D. and I. Titov. (2020). "Graph Convolutions over Constituent Trees for Syntax-Aware Semantic Role Labeling". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 3915–3928.

Marquez, L., P. Comas, J. Giménez, and N. Catala. (2005). "Semantic role labeling as sequential tagging". In: *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*. 193–196.

Miao, Y., L. Yu, and P. Blunsom. (2016). "Neural variational inference for text processing". In: *International conference on machine learning*. PMLR. 1727–1736.

Mihalcea, R. (2005). "Unsupervised large-vocabulary word sense disambiguation with graph-based algorithms for sequence data labeling". In: *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. 411–418.

Mihalcea, R. and D. Radev. (2011). *Graph-based natural language processing and information retrieval*. Cambridge university press.

Mihalcea, R. and P. Tarau. (2004). "Textrank: Bringing order into text". In: *Proceedings of the 2004 conference on empirical methods in natural language processing*. 404–411.

Mihaylov, T., P. Clark, T. Khot, and A. Sabharwal. (2018). "Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2381–2391.

Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. (2013). "Distributed Representations of Words and Phrases and their Compositionality". In: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*. Ed. by C. J. C. Burges, L. Bottou, Z. Ghahramani, and K. Q. Weinberger. 3111–3119. URL: https://proceedings.neurips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html.

Miller, A., A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston. (2016). "Key-Value Memory Networks for Directly Reading Documents". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics. 1400–1409. DOI: 10.18653/v1/D16-1147. URL: https://www.aclweb.org/anthology/D16-1147.

Minkov, E., W. W. Cohen, and A. Y. Ng. (2006). "Contextual search and name disambiguation in email using graphs". In: *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. 27–34.

Monz, C. and B. J. Dorr. (2005). "Iterative translation disambiguation for cross-language information retrieval". In: *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*. 520–527.

Nair, V. and G. E. Hinton. (2010). "Rectified linear units improve restricted boltzmann machines". In: *ICML*.

Nallapati, R., B. Zhou, C. dos Santos, Ç. Gulçehre, and B. Xiang. (2016). "Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond". In: *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. 280–290.

Nathani, D., J. Chauhan, C. Sharma, and M. Kaul. (2019a). "Learning Attention-based Embeddings for Relation Prediction in Knowledge Graphs". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics. 4710–4723. DOI: 10.18653/v1/P19-1466. URL: https://www.aclweb.org/anthology/P19-1466.

Nathani, D., J. Chauhan, C. Sharma, and M. Kaul. (2019b). "Learning Attention-based Embeddings for Relation Prediction in Knowledge Graphs". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 4710–4723.

Nguyen, T. H. and R. Grishman. (2018). "Graph convolutional networks with argument-aware pooling for event detection". In: *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*. AAAI press. 5900–5907.

Niu, Z.-Y., D. Ji, and C. L. Tan. (2005). "Word sense disambiguation using label propagation based semi-supervised learning". In: *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*. 395–402.

Nivre et al., J. (2018). "Universal Dependencies 2.2". URL: http://hdl.handle.net/11234/1-2837.

Norcliffe-Brown, W., S. Vafeias, and S. Parisot. (2018). "Learning Conditioned Graph Structures for Interpretable Visual Question Answering". In: *Advances in neural information processing systems*. 8334–8343.

Page, L., S. Brin, R. Motwani, and T. Winograd. (1999). "The PageRank citation ranking: Bringing order to the web." *Tech. rep.* Stanford InfoLab.

Palangi, H., L. Deng, Y. Shen, J. Gao, X. He, J. Chen, X. Song, and R. Ward. (2016). "Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval". *IEEE/ACM Transactions on Audio, Speech, and Language Processing*. 24(4): 694–707.

Pan, L., Y. Xie, Y. Feng, T.-S. Chua, and M.-Y. Kan. (2020). "Semantic Graphs for Generating Deep Questions". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 1463–1475.

Pang, B. and L. Lee. (2004). "A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts". In: *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*. 271–278.

Pang, B., L. Lee, and S. Vaithyanathan. (2002). "Thumbs up? Sentiment Classification using Machine Learning Techniques". In: *EMNLP*.

Pang, L., Y. Lan, J. Guo, J. Xu, S. Wan, and X. Cheng. (2016). "Text matching as image recognition". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 30. No. 1.

Paulus, R., C. Xiong, and R. Socher. (2018). "A Deep Reinforced Model for Abstractive Summarization". In: *International Conference on Learning Representations*.

Peng, H., J. Li, Y. He, Y. Liu, M. Bao, L. Wang, Y. Song, and Q. Yang. (2018). "Large-scale hierarchical text classification with recursively regularized deep graph-cnn". In: *Proceedings of the 2018 world wide web conference*. 1063–1072.

Peng, H., N. Pappas, D. Yogatama, R. Schwartz, N. Smith, and L. Kong. (2021). "Random Feature Attention". In: *International Conference on Learning Representations*.

Pennington, J., R. Socher, and C. D. Manning. (2014). "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

Peters, M., M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. (2018). "Deep Contextualized Word Representations". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. 2227–2237.

Phan, X.-H., L.-M. Nguyen, and S. Horiguchi. (2008). "Learning to classify short and sparse text & web with hidden topics from large-scale data collections". In: *Proceedings of the 17th international conference on World Wide Web*. 91–100.

Ponte, J. M. and W. B. Croft. (1998). "A language modeling approach to information retrieval". In: *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. 275–281.

Pontiki, M., D. Galanis, H. Papageorgiou, I. Androutsopoulos, S. Manandhar, M. Al-Smadi, M. Al-Ayyoub, Y. Zhao, B. Qin, O. De Clercq, *et al.* (2016). "Semeval-2016 task 5: Aspect based sentiment analysis". In: *International workshop on semantic evaluation*. 19–30.

Pontiki, M., D. Galanis, H. Papageorgiou, S. Manandhar, and I. Androutsopoulos. (2015). "Semeval-2015 task 12: Aspect based sentiment analysis". In: *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*. 486–495.

Pontiki, M., D. Galanis, J. Pavlopoulos, H. Papageorgiou, I. Androutsopoulos, and S. Manandhar. (2014). "SemEval-2014 Task 4: Aspect Based Sentiment Analysis". In: *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Dublin, Ireland: Association for Computational Linguistics. 27–35. DOI: 10.3115/v1/S14-2004. URL: https://www.aclweb.org/anthology/S14-2004.

Pouran Ben Veyseh, A., N. Nouri, F. Dernoncourt, Q. H. Tran, D. Dou, and T. H. Nguyen. (2020). "Improving Aspect-based Sentiment Analysis with Gated Graph Convolutional Networks and Syntax-based Regulation". In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics. 4543–4548. DOI: 10.18653/v1/2020.findings-emnlp.407. URL: https://www.aclweb.org/anthology/2020.findings-emnlp.407 (accessed on 12/26/2020).

Pourdamghani, N., Y. Gao, U. Hermjakob, and K. Knight. (2014). "Aligning English Strings with Abstract Meaning Representation Graphs". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics. 425–429. DOI: 10.3115/v1/D14-1048. URL: https://www.aclweb.org/anthology/D14-1048.

Pourdamghani, N., K. Knight, and U. Hermjakob. (2016). "Generating english from abstract meaning representations". In: *Proceedings of the 9th international natural language generation conference*. 21–25.

Qian, Y., E. Santus, Z. Jin, J. Guo, and R. Barzilay. (2019). "GraphIE: A Graph-Based Framework for Information Extraction". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 751–761.

Qiu, J., Q. Chen, Y. Dong, J. Zhang, H. Yang, M. Ding, K. Wang, and J. Tang. (2020). "Gcc: Graph contrastive coding for graph neural network pre-training". In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1150–1160.

Qiu, L., Y. Xiao, Y. Qu, H. Zhou, L. Li, W. Zhang, and Y. Yu. (2019). "Dynamically Fused Graph Network for Multi-hop Reasoning". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics. 6140–6150. DOI: 10.18653/v1/P19-1617. URL: https://www.aclweb.org/anthology/P19-1617.

Qu, M., T. Gao, L.-P. Xhonneux, and J. Tang. (2020). "Few-shot Relation Extraction via Bayesian Meta-learning on Relation Graphs". In: *International Conference on Machine Learning*. PMLR. 7867–7876.

Radford, A., J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. (2019). "Language models are unsupervised multitask learners". *OpenAI blog*. 1(8): 9.

Rajpurkar, P., J. Zhang, K. Lopyrev, and P. Liang. (2016). "SQuAD: 100, 000+ Questions for Machine Comprehension of Text". In: *EMNLP*.

Ramage, D., A. N. Rafferty, and C. D. Manning. (2009). "Random walks for text semantic similarity". In: *Proceedings of the 2009 workshop on graph-based methods for natural language processing (TextGraphs-4)*. 23–31.

Ran, Q., Y. Lin, P. Li, J. Zhou, and Z. Liu. (2019). "NumNet: Machine Reading Comprehension with Numerical Reasoning". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*. Ed. by K. Inui, J. Jiang, V. Ng, and X. Wan. Association for Computational Linguistics. 2474–2484.

Rashkin, H., M. Sap, E. Allaway, N. A. Smith, and Y. Choi. (2018). "Event2mind: Commonsense inference on events, intents, and reactions". *arXiv preprint arXiv:1805.06939*.

Reddy, S., D. Chen, and C. D. Manning. (2019). "Coqa: A conversational question answering challenge". *Transactions of the Association for Computational Linguistics*. 7: 249–266.

Redmon, J. and A. Farhadi. (2018). "Yolov3: An incremental improvement". *arXiv preprint arXiv:1804.02767*.

Ribeiro, L. F. R., C. Gardent, and I. Gurevych. (2019a). "Enhancing AMR-to-Text Generation with Dual Graph Representations". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics. 3183–3194. DOI: 10.18653/v1/D19-1314. URL: https://www.aclweb.org/anthology/D19-1314.

Ribeiro, L. F., C. Gardent, and I. Gurevych. (2019b). "Enhancing amr-to-text generation with dual graph representations". *arXiv preprint arXiv:1909.00352*.

Riedel, S., L. Yao, and A. McCallum. (2010). "Modeling relations and their mentions without labeled text". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 148–163.

Sachan, D. S., L. Wu, M. Sachan, and W. Hamilton. (2020). "Stronger Transformers for Neural Multi-Hop Question Generation". *arXiv preprint arXiv:2010.11374*.

Sagae, K. and A. Lavie. (2005). "A Classifier-Based Parser with Linear Run-Time Complexity". In: *Proceedings of the Ninth International Workshop on Parsing Technology*. Vancouver, British Columbia: Association for Computational Linguistics. 125–132. URL: https://www.aclweb.org/anthology/W05-1513.

Sahu, S. K., F. Christopoulou, M. Miwa, and S. Ananiadou. (2019). "Inter-sentence Relation Extraction with Document-level Graph Convolutional Neural Network". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics. 4309–4316. DOI: 10.18653/v1/P19-1423. URL: https://www.aclweb.org/anthology/P19-1423.

Sandhaus, E. (2008). "The New York Times Annotated Corpus". Version V1. DOI: 11272.1/AB2/GZC6PL. URL: https://hdl.handle.net/11272.1/AB2/GZC6PL.

Santoro, A., D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. (2017). "A simple neural network module for relational reasoning". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 4974–4983.

Saxena, A., A. Tripathi, and P. Talukdar. (2020). "Improving Multi-hop Question Answering over Knowledge Graphs using Knowledge Base Embeddings". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics. 4498–4507. DOI: 10.18653/v1/2020.acl-main.412. URL: https://www.aclweb.org/anthology/2020.acl-main.412.

Schlichtkrull, M., T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling. (2018). "Modeling relational data with graph convolutional networks". In: *European Semantic Web Conference*. Springer. 593–607.

Schuster, M. and K. K. Paliwal. (1997). "Bidirectional recurrent neural networks". *IEEE Transactions on Signal Processing*. 45(11): 2673–2681. DOI: 10.1109/78.650093.

See, A., P. J. Liu, and C. D. Manning. (2017). "Get To The Point: Summarization with Pointer-Generator Networks". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1073–1083.

Seo, M. J., A. Kembhavi, A. Farhadi, and H. Hajishirzi. (2017). "Bidirectional Attention Flow for Machine Comprehension". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. URL: https://openreview.net/forum?id=HJ0UKP9ge.

Serban, I., A. Sordoni, Y. Bengio, A. Courville, and J. Pineau. (2016). "Building end-to-end dialogue systems using generative hierarchical neural network models". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 30. No. 1.

Serban, I., A. Sordoni, R. Lowe, L. Charlin, J. Pineau, A. Courville, and Y. Bengio. (2017). "A hierarchical latent variable encoder-decoder model for generating dialogues". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 31. No. 1.

Shang, C., Y. Tang, J. Huang, J. Bi, X. He, and B. Zhou. (2019). "End-to-end structure-aware convolutional networks for knowledge base completion". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 3060–3067.

Shao, B., Y. Gong, W. Qi, G. Cao, J. Ji, and X. Lin. (2020). "Graph-Based Transformer with Cross-Candidate Verification for Semantic Parsing". *Proceedings of the AAAI Conference on Artificial Intelligence*. 34(05): 8807–8814. DOI: 10.1609/aaai.v34i05.6408. URL: https://ojs.aaai.org/index.php/AAAI/article/view/6408.

Shaw, P., J. Uszkoreit, and A. Vaswani. (2018). "Self-Attention with Relative Position Representations". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. New Orleans, Louisiana: Association for Computational Linguistics. 464–468. DOI: 10.18653/v1/N18-2074. URL: https://www.aclweb.org/anthology/N18-2074.

Shen, D. and M. Lapata. (2007). "Using semantic roles to improve question answering". In: *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*. 12–21.

Shen, Z., M. Zhang, H. Zhao, S. Yi, and H. Li. (2021). "Efficient attention: Attention with linear complexities". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 3531–3539.

Shi, C., M. Xu, H. Guo, M. Zhang, and J. Tang. (2020). "A graph to graphs framework for retrosynthesis prediction". In: *International Conference on Machine Learning*. PMLR. 8818–8827.

Simonovsky, M. and N. Komodakis. (2017). "Dynamic edge-conditioned filters in convolutional neural networks on graphs". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3693–3702.

Song, L., D. Gildea, Y. Zhang, Z. Wang, and J. Su. (2019). "Semantic neural machine translation using AMR". *Transactions of the Association for Computational Linguistics*. 7: 19–31.

Song, L., X. Peng, Y. Zhang, Z. Wang, and D. Gildea. (2017). "AMR-to-text Generation with Synchronous Node Replacement Grammar". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 7–13.

Song, L., A. Wang, J. Su, Y. Zhang, K. Xu, Y. Ge, and D. Yu. (2020). "Structural Information Preserving for Graph-to-Text Generation". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics. 7987–7998. DOI: 10.18653/v1/2020.acl-main.712. URL: https://www.aclweb.org/anthology/2020.acl-main.712.

Song, L., Z. Wang, W. Hamza, Y. Zhang, and D. Gildea. (2018a). "Leveraging context information for natural question generation". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. 569–574.

Song, L., Z. Wang, M. Yu, Y. Zhang, R. Florian, and D. Gildea. (2018b). "Exploring graph-structured passage representation for multi-hop reading comprehension with graph neural networks". *arXiv preprint arXiv:1809.02040*.

Song, L., Y. Zhang, Z. Wang, and D. Gildea. (2018c). "A Graph-to-Sequence Model for AMR-to-Text Generation". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics. 1616–1626. DOI: 10.18653/v1/P18-1150. URL: https://www.aclweb.org/anthology/P18-1150.

Song, L., Y. Zhang, Z. Wang, and D. Gildea. (2018d). "A graph-to-sequence model for amr-to-text generation". *arXiv preprint arXiv:1805.02473*.

Song, L., Y. Zhang, Z. Wang, and D. Gildea. (2018e). "N-ary Relation Extraction using Graph-State LSTM". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics. 2226–2235. DOI: 10.18653/v1/D18-1246. URL: https://www.aclweb.org/anthology/D18-1246.

Sorokin, D. and I. Gurevych. (2018a). "Modeling Semantics with Gated Graph Neural Networks for Knowledge Base Question Answering". In: *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*. Ed. by E. M. Bender, L. Derczynski, and P. Isabelle. Association for Computational Linguistics. 3306–3317.

Sorokin, D. and I. Gurevych. (2018b). "Modeling semantics with gated graph neural networks for knowledge base question answering". *arXiv preprint arXiv:1808.04126*.

Speer, R., J. Chin, and C. Havasi. (2017). "Conceptnet 5.5: An open multilingual graph of general knowledge". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 31. No. 1.

Su, D., Y. Xu, W. Dai, Z. Ji, T. Yu, and P. Fung. (2020). "Multi-hop Question Generation with Graph Convolutional Network". In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics. 4636–4647. DOI: 10.18653/v1/2020.findings-emnlp.416. URL: https://www.aclweb.org/anthology/2020.findings-emnlp.416.

Suchanek, F. M., G. Kasneci, and G. Weikum. (2008). "Yago: A large ontology from wikipedia and wordnet". *Journal of Web Semantics*. 6(3): 203–217.

Sui, D., Y. Chen, K. Liu, J. Zhao, and S. Liu. (2019). "Leverage lexical knowledge for chinese named entity recognition via collaborative graph network". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 3821–3831.

Sukhbaatar, S., A. Szlam, J. Weston, and R. Fergus. (2015). "End-to-end memory networks". In: *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*. 2440–2448.

Sun, C., Y. Gong, Y. Wu, M. Gong, D. Jiang, M. Lan, S. Sun, and N. Duan. (2019a). "Joint Type Inference on Entities and Relations via Graph Convolutional Networks". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics. 1361–1370. DOI: 10.18653/v1/P19-1131. URL: https://www.aclweb.org/anthology/P19-1131.

Sun, H., T. Bedrax-Weiss, and W. Cohen. (2019b). "PullNet: Open Domain Question Answering with Iterative Retrieval on Knowledge Bases and Text". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics. 2380–2390. DOI: 10.18653/v1/D19-1242. URL: https://www.aclweb.org/anthology/D19-1242.

Sun, H., B. Dhingra, M. Zaheer, K. Mazaitis, R. Salakhutdinov, and W. Cohen. (2018a). "Open Domain Question Answering Using Early Fusion of Knowledge Bases and Text". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics. 4231–4242. DOI: 10.18653/v1/D18-1455. URL: https://www.aclweb.org/anthology/D18-1455.

Sun, K., R. Zhang, S. Mensah, Y. Mao, and X. Liu. (2019c). "Aspect-Level Sentiment Analysis Via Convolution over Dependency Tree". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics. 5679–5688. DOI: 10.18653/v1/D19-1569. URL: https://www.aclweb.org/anthology/D19-1569.

Sun, T., Y. Shao, X. Qiu, Q. Guo, Y. Hu, X. Huang, and Z. Zhang. (2020a). "CoLAKE: Contextualized Language and Knowledge Embedding". In: *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*. International Committee on Computational Linguistics. 3660–3670.

Sun, Y., J. Han, X. Yan, P. S. Yu, and T. Wu. (2011). "Pathsim: Meta path-based top-k similarity search in heterogeneous information networks". *Proceedings of the VLDB Endowment*. 4(11): 992–1003.

Sun, Z., W. Hu, and C. Li. (2017). "Cross-lingual entity alignment via joint attribute-preserving embedding". In: *International Semantic Web Conference*. Springer. 628–644.

Sun, Z., W. Hu, Q. Zhang, and Y. Qu. (2018b). "Bootstrapping Entity Alignment with Knowledge Graph Embedding." In: *IJCAI*. Vol. 18. 4396–4402.

Sun, Z., C. Wang, W. Hu, M. Chen, J. Dai, W. Zhang, and Y. Qu. (2020b). "Knowledge graph alignment network with gated multi-hop neighborhood aggregation". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. No. 01. 222–229.

Sun, Z., Q. Zhu, Y. Xiong, Y. Sun, L. Mou, and L. Zhang. (2020c). "Treegen: A tree-based transformer architecture for code generation". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. No. 05. 8984–8991.

Sutskever, I., O. Vinyals, and Q. V. Le. (2014). "Sequence to Sequence Learning with Neural Networks". In: *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger. 3104–3112. URL: https://proceedings.neurips.cc/paper/2014/hash/a14ac55a4f27472c5d894ec1c3c743d2-Abstract.html.

Tai, K. S., R. Socher, and C. D. Manning. (2015). "Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 1556–1566.

Talmor, A. and J. Berant. (2018). "The Web as a Knowledge-Base for Answering Complex Questions". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. 641–651.

Talmor, A., J. Herzig, N. Lourie, and J. Berant. (2019). "CommonsenseQA: A Question Answering Challenge Targeting Commonsense Knowledge". In: *NAACL-HLT (1)*.

Tang, D., B. Qin, X. Feng, and T. Liu. (2016). "Effective LSTMs for Target-Dependent Sentiment Classification". In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. 3298–3307.

Tang, H., D. Ji, C. Li, and Q. Zhou. (2020a). "Dependency Graph Enhanced Dual-transformer Structure for Aspect-based Sentiment Classification". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics. 6578–6588. DOI: 10.18653/v1/2020.acl-main.588. URL: https://www.aclweb.org/anthology/2020.acl-main.588.

Tang, J., M. Qu, and Q. Mei. (2015). "Pte: Predictive text embedding through large-scale heterogeneous text networks". In: *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 1165–1174.

Tang, Z., Y. Shen, X. Ma, W. Xu, J. Yu, and W. Lu. (2020b). "Multi-hop Reading Comprehension across Documents with Path-based Graph Convolutional Network". In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*. Ed. by C. Bessiere. ijcai.org. 3905–3911.

Tang, Z., Y. Shen, X. Ma, W. Xu, J. Yu, and W. Lu. (2020c). "Multi-hop reading comprehension across documents with path-based graph convolutional network". *arXiv preprint arXiv:2006.06478*.

Tarau, P., R. Mihalcea, and E. Figa. (2005). "Semantic document engineering with WordNet and PageRank". In: *Proceedings of the 2005 ACM symposium on Applied computing*. 782–786.

Taylor, A., M. Marcus, and B. Santorini. (2003). "The Penn treebank: an overview". *Treebanks*: 5–22.

Teru, K., E. Denis, and W. Hamilton. (2020). "Inductive relation prediction by subgraph reasoning". In: *International Conference on Machine Learning*. PMLR. 9448–9457.

Thayaparan, M., M. Valentino, V. Schlegel, and A. Freitas. (2019). "Identifying Supporting Facts for Multi-hop Question Answering with Document Graph Networks". In: *Proceedings of the Thirteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-13)*. Hong Kong: Association for Computational Linguistics. 42–51. DOI: 10.18653/v1/D19-5306. URL: https://www.aclweb.org/anthology/D19-5306.

Thompson, A. (2017). "All the news: 143,000 articles from 15 american publications".

Toutanova, K., D. Chen, P. Pantel, H. Poon, P. Choudhury, and M. Gamon. (2015). "Representing text for joint embedding of text and knowledge bases". In: *Proceedings of the 2015 conference on empirical methods in natural language processing*. 1499–1509.

Trischler, A., T. Wang, X. Yuan, J. Harris, A. Sordoni, P. Bachman, and K. Suleman. (2017). "NewsQA: A Machine Comprehension Dataset". In: *Proceedings of the 2nd Workshop on Representation Learning for NLP*. 191–200.

Trouillon, T., J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard. (2016). "Complex embeddings for simple link prediction". In: *International Conference on Machine Learning*. PMLR. 2071–2080.

Tsai, Y.-H. H., S. Bai, M. Yamada, L.-P. Morency, and R. Salakhutdinov. (2019). "Transformer Dissection: An Unified Understanding for Transformer's Attention via the Lens of Kernel". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 4335–4344.

Tu, M., G. Wang, J. Huang, Y. Tang, X. He, and B. Zhou. (2019a). "Multi-hop Reading Comprehension across Multiple Documents by Reasoning over Heterogeneous Graphs". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics. 2704–2713. DOI: 10.18653/v1/P19-1260. URL: https://www.aclweb.org/anthology/P19-1260.

Tu, M., G. Wang, J. Huang, Y. Tang, X. He, and B. Zhou. (2019b). "Multi-hop Reading Comprehension across Multiple Documents by Reasoning over Heterogeneous Graphs". In: *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. Ed. by A. Korhonen, D. R. Traum, and L. Màrquez. Association for Computational Linguistics. 2704–2713.

Tu, Z., Z. Lu, Y. Liu, X. Liu, and H. Li. (2016). "Modeling coverage for neural machine translation". *arXiv preprint arXiv:1601.04811*.

Usbeck, R., A.-C. N. Ngomo, B. Haarmann, A. Krithara, M. Röder, and G. Napolitano. (2017). "7th open challenge on question answering over linked data (QALD-7)". In: *Semantic web evaluation challenge*. Springer. 59–69.

Vashishth, S., R. Joshi, S. S. Prayaga, C. Bhattacharyya, and P. Talukdar. (2018). "RESIDE: Improving Distantly-Supervised Neural Relation Extraction using Side Information". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics. 1257–1266. DOI: 10.18653/v1/D18-1157. URL: https://www.aclweb.org/anthology/D18-1157.

Vashishth, S., S. Sanyal, V. Nitin, and P. Talukdar. (2019). "Composition-based multi-relational graph convolutional networks". *arXiv preprint arXiv:1911.03082*.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. (2017). "Attention is All you Need". In: *NIPS*.

Velickovic, P., L. Buesing, M. C. Overlan, R. Pascanu, O. Vinyals, and C. Blundell. (2020). "Pointer Graph Networks". In: *Advances in Neural Information Processing Systems*.

Velickovic, P., G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. (2018). "Graph Attention Networks". In: *6th International Conference on Learning Representations*.

Vinyals, O., S. Bengio, and M. Kudlur. (2016). "Order Matters: Sequence to sequence for sets". In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Ed. by Y. Bengio and Y. LeCun. URL: http://arxiv.org/abs/1511.06391.

Vinyals, O., M. Fortunato, and N. Jaitly. (2015). "Pointer networks". In: *Advances in Neural Information Processing Systems*. 2692–2700.

Vitale, D., P. Ferragina, and U. Scaiella. (2012). "Classification of short texts by deploying topical annotations". In: *European Conference on Information Retrieval*. Springer. 376–387.

Wan, S., Y. Lan, J. Guo, J. Xu, L. Pang, and X. Cheng. (2016). "A deep architecture for semantic matching with multiple positional sentence representations". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 30. No. 1.

Wang, D., P. Liu, Y. Zheng, X. Qiu, and X. Huang. (2020a). "Heterogeneous Graph Neural Networks for Extractive Document Summarization". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics. 6209–6219. DOI: 10.18653/v1/2020.acl-main.553. URL: https://www.aclweb.org/anthology/2020.acl-main.553.

Wang, H., S. Li, R. Pan, and M. Mao. (2019a). "Incorporating Graph Attention Mechanism into Knowledge Graph Reasoning Based on Deep Reinforcement Learning". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics. 2623–2631. DOI: 10.18653/v1/D19-1264. URL: https://www.aclweb.org/anthology/D19-1264.

Wang, K., W. Shen, Y. Yang, X. Quan, and R. Wang. (2020b). "Relational Graph Attention Network for Aspect-based Sentiment Analysis". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics. 3229–3238. DOI: 10.18653/v1/2020.acl-main.295. URL: https://www.aclweb.org/anthology/2020.acl-main.295 (accessed on 12/26/2020).

Wang, L., Z. Xu, Z. Lin, H. Zheng, and Y. Shen. (2020c). "Answer-driven Deep Question Generation based on Reinforcement Learning". In: *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics. 5159–5170. DOI: 10.18653/v1/2020.coling-main.452. URL: https://www.aclweb.org/anthology/2020.coling-main.452.

Wang, L., Z. Xu, Z. Lin, H. Zheng, and Y. Shen. (2020d). "Answer-driven Deep Question Generation based on Reinforcement Learning". In: *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*. Ed. by D. Scott, N. Bel, and C. Zong. International Committee on Computational Linguistics. 5159–5170.

Wang, M., L. Yu, D. Zheng, Q. Gan, Y. Gai, Z. Ye, M. Li, J. Zhou, Q. Huang, C. Ma, *et al.* (2019b). "Deep Graph Library: Towards Efficient and Scalable Deep Learning on Graphs."

Wang, P., J. Han, C. Li, and R. Pan. (2019c). "Logic attention based neighborhood aggregation for inductive knowledge graph embedding". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 7152–7159.

Wang, R., D. Zhou, and Y. He. (2019d). "Atm: Adversarial-neural topic model". *Information Processing & Management*. 56(6): 102098.

Wang, S., B. Z. Li, M. Khabsa, H. Fang, and H. Ma. (2020e). "Linformer: Self-Attention with Linear Complexity". *CoRR*. abs/2006.04768. arXiv: 2006.04768. URL: https://arxiv.org/abs/2006.04768.

Wang, T., X. Wan, and H. Jin. (2020f). "AMR-To-Text Generation with Graph Transformer". *Transactions of the Association for Computational Linguistics*. 8: 19–33.

Wang, T., X. Wan, and S. Yao. (2020g). "Better AMR-To-Text Generation with Graph Structure Reconstruction". In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*. Ed. by C. Bessiere. International Joint Conferences on Artificial Intelligence Organization. 3919–3925. DOI: 10.24963/ijcai.2020/542. URL: https://doi.org/10.24963/ijcai.2020/542.

Wang, X., H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu. (2019e). "Heterogeneous graph attention network". In: *The World Wide Web Conference*. 2022–2032.

Wang, X., P. Kapanipathi, R. Musa, M. Yu, K. Talamadupula, I. Abdelaziz, M. Chang, A. Fokoue, B. Makni, N. Mattei, and M. Witbrock. (2019f). "Improving Natural Language Inference Using External Knowledge in the Science Questions Domain". *Proceedings of the AAAI Conference on Artificial Intelligence*. 33(01): 7208–7215. DOI: 10.1609/aaai.v33i01.33017208. URL: https://ojs.aaai.org/index.php/AAAI/article/view/4705.

Wang, Y., X. Liu, and S. Shi. (2017). "Deep neural solver for math word problems". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 845–854.

Wang, Z., Q. Lv, X. Lan, and Y. Zhang. (2018). "Cross-lingual knowledge graph alignment via graph convolutional networks". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 349–357.

Wang, Z., J. Yang, and X. Ye. (2020h). "Knowledge Graph Alignment with Entity-Pair Embedding". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics. 1672–1680. DOI: 10.18653/v1/2020.emnlp-main.130. URL: https://www.aclweb.org/anthology/2020.emnlp-main.130.

Wang, Z., Z. Ren, C. He, P. Zhang, and Y. Hu. (2019g). "Robust Embedding with Multi-Level Structures for Link Prediction." In: *IJCAI*. 5240–5246.

Welbl, J., P. Stenetorp, and S. Riedel. (2018). "Constructing datasets for multi-hop reading comprehension across documents". *Transactions of the Association for Computational Linguistics*. 6: 287–302.

Williams, A., N. Nangia, and S. Bowman. (2018). "A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference". *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. DOI: 10.18653/v1/n18-1101. URL: http://dx.doi.org/10.18653/v1/N18-1101.

Williams, J. D., M. Henderson, A. Raux, B. Thomson, A. Black, and D. Ramachandran. (2014). "The dialog state tracking challenge series". *AI Magazine*. 35(4): 121–124.

Williams, R. J. (1992). "Simple statistical gradient-following algorithms for connectionist reinforcement learning". *Machine learning*. 8(3-4): 229–256.

Wu, J., M. Cao, J. C. K. Cheung, and W. L. Hamilton. (2020a). "TeMP: Temporal Message Passing for Temporal Knowledge Graph Completion". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics. 5730–5746. DOI: 10.18653/v1/2020.emnlp-main.462. URL: https://www.aclweb.org/anthology/2020.emnlp-main.462.

Wu, L., P. Cui, J. Pei, and L. Zhao. (2022). *Graph Neural Networks: Foundations, Frontiers, and Applications*. Singapore: Springer Singapore. 725.

Wu, Q., Q. Zhang, J. Fu, and X. Huang. (2020b). "A Knowledge-Aware Sequence-to-Tree Network for Math Word Problem Solving". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics. 7137–7146. DOI: 10.18653/v1/2020.emnlp-main.579. URL: https://www.aclweb.org/anthology/2020.emnlp-main.579.

Wu, Y., X. Liu, Y. Feng, Z. Wang, R. Yan, and D. Zhao. (2019a). "Relation-aware entity alignment for heterogeneous knowledge graphs". In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press. 5278–5284.

Wu, Y., X. Liu, Y. Feng, Z. Wang, and D. Zhao. (2019b). "Jointly Learning Entity and Relation Representations for Entity Alignment". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 240–249.

Wu, Z., R. Koncel-Kedziorski, M. Ostendorf, and H. Hajishirzi. (2020c). "Extracting Summary Knowledge Graphs from Long Documents". *arXiv preprint arXiv:2009.09162*.

Xia, M., G. Huang, L. Liu, and S. Shi. (2019). "Graph Based Translation Memory for Neural Machine Translation". *Proceedings of the AAAI Conference on Artificial Intelligence*. 33(01): 7297–7304. DOI: 10.1609/aaai.v33i01.33017297. URL: https://ojs.aaai.org/index.php/AAAI/article/view/4716.

Xia, Q., R. Wang, Z. Li, Y. Zhang, and M. Zhang. (2020). "Semantic Role Labeling with Heterogeneous Syntactic Knowledge". In: *Proceedings of the 28th International Conference on Computational Linguistics*. 2979–2990.

Xiao, F., J. Li, H. Zhao, R. Wang, and K. Chen. (2019). "Lattice-Based Transformer Encoder for Neural Machine Translation". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics. 3090–3097. DOI: 10.18653/v1/P19-1298. URL: https://www.aclweb.org/anthology/P19-1298.

Xie, Z., G. Zhou, J. Liu, and X. Huang. (2020). "ReInceptionE: Relation-aware inception network with joint local-global structural information for knowledge graph embedding". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 5929–5939.

Xiong, C., V. Zhong, and R. Socher. (2017a). "Dynamic Coattention Networks For Question Answering". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. URL: https://openreview.net/forum?id=rJeKjwvclx.

Xiong, W., T. Hoang, and W. Y. Wang. (2017b). "DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 564–573.

Xu, J., Z. Gan, Y. Cheng, and J. Liu. (2020a). "Discourse-aware neural extractive text summarization". In: *Proceedings of the 58th annual meeting of the association for computational linguistics*. 5021–5031.

Xu, K., L. Song, Y. Feng, Y. Song, and D. Yu. (2020b). "Coordinated Reasoning for Cross-Lingual Knowledge Graph Alignment". *arXiv preprint arXiv:2001.08728*.

Xu, K., L. Wang, M. Yu, Y. Feng, Y. Song, Z. Wang, and D. Yu. (2019a). "Cross-lingual Knowledge Graph Alignment via Graph Matching Neural Network". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics. 3156–3161. DOI: 10.18653/v1/P19-1304. URL: https://www.aclweb.org/anthology/P19-1304.

Xu, K., L. Wu, Z. Wang, Y. Feng, and V. Sheinin. (2018a). "SQL-to-Text Generation with Graph-to-Sequence Model". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics. 931–936. DOI: 10.18653/v1/D18-1112. URL: https://www.aclweb.org/anthology/D18-1112.

Xu, K., L. Wu, Z. Wang, Y. Feng, M. Witbrock, and V. Sheinin. (2018b). "Graph2seq: Graph to sequence learning with attention-based neural networks". *arXiv preprint arXiv:1804.00823*.

Xu, K., L. Wu, Z. Wang, M. Yu, L. Chen, and V. Sheinin. (2018c). "Exploiting Rich Syntactic Information for Semantic Parsing with Graph-to-Sequence Model". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics. 918–924. DOI: 10.18653/v1/D18-1110. URL: https://www.aclweb.org/anthology/D18-1110.

Xu, M., L. Li, D. F. Wai, Q. Liu, and L. S. Chao. (2020c). "Document Graph for Neural Machine Translation". *ArXiv*. abs/2012.03477.

Xu, X., W. Feng, Y. Jiang, X. Xie, Z. Sun, and Z.-H. Deng. (2019b). "Dynamically Pruned Message Passing Networks for Large-Scale Knowledge Graph Reasoning". *arXiv preprint arXiv:1909.11334*.

Yan, H., X. Jin, X. Meng, J. Guo, and X. Cheng. (2019). "Event Detection with Multi-Order Graph Convolution and Aggregated Attention". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics. 5766–5770. DOI: 10.18653/v1/D19-1582. URL: https://www.aclweb.org/anthology/D19-1582.

Yang, B., W.-t. Yih, X. He, J. Gao, and L. Deng. (2014). "Embedding entities and relations for learning and inference in knowledge bases". *arXiv preprint arXiv:1412.6575*.

Yang, H.-W., Y. Zou, P. Shi, W. Lu, J. Lin, and S. Xu. (2019). "Aligning Cross-Lingual Entities with Multi-Aspect Information". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 4422–4432.

Yang, K. and J. Deng. (2020). "Strongly Incremental Constituency Parsing with Graph Neural Networks". *arXiv preprint arXiv:2010.14568*.

Yang, L., F. Wu, J. Gu, C. Wang, X. Cao, D. Jin, and Y. Guo. (2020). "Graph Attention Topic Modeling Network". In: *Proceedings of The Web Conference 2020. WWW '20*. New York, NY, USA: Association for Computing Machinery. 144–154. ISBN: 978-1-4503-7023-3. DOI: 10.1145/3366423.3380102. URL: http://doi.org/10.1145/3366423.3380102 (accessed on 12/25/2020).

Yang, L., Q. Ai, J. Guo, and W. B. Croft. (2016). "aNMM: Ranking short answer texts with attention-based neural matching model". In: *Proceedings of the 25th ACM international on conference on information and knowledge management*. 287–296.

Yang, Z., P. Qi, S. Zhang, Y. Bengio, W. Cohen, R. Salakhutdinov, and C. D. Manning. (2018a). "HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2369–2380.

Yang, Z., J. Zhao, B. Dhingra, K. He, W. W. Cohen, R. R. Salakhutdinov, and Y. LeCun. (2018b). "Glomo: Unsupervised learning of transferable relational graphs". In: *Advances in Neural Information Processing Systems*. 8950–8961.

Yao, L., C. Mao, and Y. Luo. (2019a). "Graph Convolutional Networks for Text Classification". In: *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press. 7370–7377.

Yao, L., C. Mao, and Y. Luo. (2019b). "Graph convolutional networks for text classification". In: *Proc. AAAI Conf. Artif. Intell.* Vol. 33. 7370–7377.

Yao, S., T. Wang, and X. Wan. (2020). "Heterogeneous graph transformer for graph-to-sequence learning". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 7145–7154.

Yao, T., Y. Pan, Y. Li, and T. Mei. (2018). "Exploring visual relationship for image captioning". In: *Proceedings of the European conference on computer vision (ECCV)*. 684–699.

Yasunaga, M., H. Ren, A. Bosselut, P. Liang, and J. Leskovec. (2021). "QA-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering".

Yasunaga, M., R. Zhang, K. Meelu, A. Pareek, K. Srinivasan, and D. Radev. (2017). "Graph-based Neural Multi-Document Summarization". In: *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*. Vancouver, Canada: Association for Computational Linguistics. 452–462. DOI: 10.18653/v1/K17-1045. URL: https://www.aclweb.org/anthology/K17-1045.

Ye, R., X. Li, Y. Fang, H. Zang, and M. Wang. (2019). "A Vectorized Relational Graph Convolutional Network for Multi-Relational Network Alignment." In: *IJCAI*. 4135–4141.

Yih, W.-t., M.-W. Chang, X. He, and J. Gao. (2015). "Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 1321–1331.

Yih, W.-t., M. Richardson, C. Meek, M.-W. Chang, and J. Suh. (2016). "The value of semantic parse labeling for knowledge base question answering". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 201–206.

Yin, P., G. Neubig, M. Allamanis, M. Brockschmidt, and A. L. Gaunt. (2018). "Learning to represent edits". *arXiv preprint arXiv:1810.13337*.

Yin, Y., F. Meng, J. Su, C. Zhou, Z. Yang, J. Zhou, and J. Luo. (2020). "A Novel Graph-based Multi-modal Fusion Encoder for Neural Machine Translation". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics. 3025–3035. DOI: 10.18653/v1/2020.acl-main.273. URL: https://www.aclweb.org/anthology/2020.acl-main.273.

Ying, R., J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec. (2018). "Hierarchical graph representation learning with differentiable pooling". *arXiv preprint arXiv:1806.08804*.

Yu, T., R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, Z. Zhang, and D. Radev. (2018). "Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics. 3911–3921. DOI: 10.18653/v1/D18-1425. URL: https://www.aclweb.org/anthology/D18-1425.

Yun, S., M. Jeong, R. Kim, J. Kang, and H. J. Kim. (2019). "Graph Transformer Networks". *Advances in Neural Information Processing Systems*. 32: 11983–11993.

Zellers, R., Y. Bisk, R. Schwartz, and Y. Choi. (2018). "Swag: A large-scale adversarial dataset for grounded commonsense inference". *arXiv preprint arXiv:1808.05326*.

Zeng, S., R. Xu, B. Chang, and L. Li. (2020). "Double Graph Based Reasoning for Document-level Relation Extraction". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics. 1630–1640. DOI: 10.18653/v1/2020.emnlp-main.127. URL: https://www.aclweb.org/anthology/2020.emnlp-main.127.

Zhang, B., Y. Zhang, R. Wang, Z. Li, and M. Zhang. (2020a). "Syntax-aware opinion role labeling with dependency graph convolutional networks". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 3249–3258.

Zhang, C., Q. Li, and D. Song. (2019a). "Aspect-based Sentiment Classification with Aspect-specific Graph Convolutional Networks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics. 4568–4578. DOI: 10.18653/v1/D19-1464. URL: https://www.aclweb.org/anthology/D19-1464 (accessed on 12/26/2020).

Zhang, C., D. Song, C. Huang, A. Swami, and N. V. Chawla. (2019b). "Heterogeneous graph neural network". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 793–803.

Zhang, C., D. Song, C. Huang, A. Swami, and N. V. Chawla. (2019c). "Heterogeneous Graph Neural Network". In: *KDD*.

Zhang, J., L. Wang, R. K.-W. Lee, Y. Bin, Y. Wang, J. Shao, and E.-P. Lim. (2020b). "Graph-to-Tree Learning for Solving Math Word Problems". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics. 3928–3937. DOI: 10.18653/v1/2020.acl-main.362. URL: https://www.aclweb.org/anthology/2020.acl-main.362.

Zhang, M. and T. Qian. (2020). "Convolution over Hierarchical Syntactic and Lexical Graphs for Aspect Level Sentiment Analysis". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics. 3540–3549. DOI: 10.18653/v1/2020.emnlp-main.286. URL: https://www.aclweb.org/anthology/2020.emnlp-main.286.

Zhang, N., S. Deng, J. Li, X. Chen, W. Zhang, and H. Chen. (2020c). "Summarizing Chinese Medical Answer with Graph Convolution Networks and Question-focused Dual Attention". In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics. 15–24. DOI: 10.18653/v1/2020.findings-emnlp.2. URL: https://www.aclweb.org/anthology/2020.findings-emnlp.2.

Zhang, N., S. Deng, Z. Sun, G. Wang, X. Chen, W. Zhang, and H. Chen. (2019d). "Long-tail Relation Extraction via Knowledge Graph Embeddings and Graph Convolution Networks". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics. 3016–3025. DOI: 10.18653/v1/N19-1306. URL: https://www.aclweb.org/anthology/N19-1306.

Zhang, S., X. Liu, J. Liu, J. Gao, K. Duh, and B. Van Durme. (2018a). "Record: Bridging the gap between human and machine commonsense reading comprehension". *arXiv preprint arXiv:1810.12885*.

Zhang, S., X. Ma, K. Duh, and B. Van Durme. (2019e). "AMR Parsing as Sequence-to-Graph Transduction". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 80–94.

Zhang, S., X. Ma, K. Duh, and B. Van Durme. (2019f). "AMR Parsing as Sequence-to-Graph Transduction". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics. 80–94. DOI: 10.18653/v1/P19-1009. URL: https://www.aclweb.org/anthology/P19-1009.

Zhang, X., J. Zhao, and Y. LeCun. (2015). "Character-level convolutional networks for text classification". *arXiv preprint arXiv:1509.01626*.

Zhang, Y., Z. Guo, Z. Teng, W. Lu, S. B. Cohen, Z. Liu, and L. Bing. (2020d). "Lightweight, Dynamic Graph Convolutional Networks for AMR-to-Text Generation". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics. 2162–2172. DOI: 10.18653/v1/2020.emnlp-main.169. URL: https://www.aclweb.org/anthology/2020.emnlp-main.169.

Zhang, Y., W. Chan, and N. Jaitly. (2017a). "Very deep convolutional networks for end-to-end speech recognition". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 4845–4849.

Zhang, Y., Q. Liu, and L. Song. (2018b). "Sentence-State LSTM for Text Representation". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 317–327.

Zhang, Y., X. Yu, Z. Cui, S. Wu, Z. Wen, and L. Wang. (2020e). "Every Document Owns Its Structure: Inductive Text Classification via Graph Neural Networks". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*. Ed. by D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault. Association for Computational Linguistics. 334–339.

Zhang, Y., X. Yu, Z. Cui, S. Wu, Z. Wen, and L. Wang. (2020f). "Every document owns its structure: Inductive text classification via graph neural networks". *arXiv preprint arXiv:2004.13826*.

Zhang, Y., P. Qi, and C. D. Manning. (2018c). "Graph Convolution over Pruned Dependency Trees Improves Relation Extraction". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics. 2205–2215. DOI: 10.18653/v1/D18-1244. URL: https://www.aclweb.org/anthology/D18-1244.

Zhang, Y., V. Zhong, D. Chen, G. Angeli, and C. D. Manning. (2017b). "Position-aware attention and supervised data improve slot filling". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 35–45.

Zhang, Y., H. Dai, Z. Kozareva, A. Smola, and L. Song. (2018d). "Variational reasoning for question answering with knowledge graph". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. No. 1.

Zhang, Z., F. Zhuang, H. Zhu, Z.-P. Shi, H. Xiong, and Q. He. (2020g). "Relational Graph Neural Network with Hierarchical Attention for Knowledge Graph Completion." In: *AAAI*. 9612–9619.

Zhao, J., X. Wang, C. Shi, B. Hu, G. Song, and Y. Ye. (2021). "Heterogeneous Graph Structure Learning for Graph Neural Networks". In: *Proceedings of the AAAI Conference on Artificial Intelligence*.

Zhao, L., W. Xu, and J. Guo. (2020a). "Improving Abstractive Dialogue Summarization with Graph Structures and Topic Words". In: *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics. 437–449. URL: https://www.aclweb.org/anthology/2020.coling-main.39.

Zhao, Y., L. Chen, Z. Chen, R. Cao, S. Zhu, and K. Yu. (2020b). "Line Graph Enhanced AMR-to-Text Generation with Mix-Order Graph Attention Networks". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics. 732–741. DOI: 10.18653/v1/2020.acl-main.67. URL: https://www.aclweb.org/anthology/2020.acl-main.67.

Zhao, Y., L. Xiang, J. Zhu, J. Zhang, Y. Zhou, and C. Zong. (2020c). "Knowledge Graph Enhanced Neural Machine Translation via Multi-task Learning on Sub-entity Granularity". In: *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics. 4495–4505. URL: https://www.aclweb.org/anthology/2020.coling-main.397.

Zheng, B., H. Wen, Y. Liang, N. Duan, W. Che, D. Jiang, M. Zhou, and T. Liu. (2020). "Document Modeling with Graph Attention Networks for Multi-grained Machine Reading Comprehension". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*. Ed. by D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault. Association for Computational Linguistics. 6708–6718.

Zheng, C. and P. Kordjamshidi. (2020). "SRLGRN: Semantic Role Labeling Graph Reasoning Network". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 8881–8891.

Zhong, V., C. Xiong, and R. Socher. (2017). "Seq2sql: Generating structured queries from natural language using reinforcement learning". *arXiv preprint arXiv:1709.00103*.

Zhou, D., X. Hu, and R. Wang. (2020a). "Neural Topic Modeling by Incorporating Document Relationship Graph". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics. 3790–3796. DOI: 10.18653/v1/2020.emnlp-main.310. URL: https://www.aclweb.org/anthology/2020.emnlp-main.310 (accessed on 12/26/2020).

Zhou, H., T. Young, M. Huang, H. Zhao, J. Xu, and X. Zhu. (2018a). "Commonsense knowledge aware conversation generation with graph attention." In: *IJCAI*. 4623–4629.

Zhou, Q., Y. Zhang, D. Ji, and H. Tang. (2020b). "AMR Parsing with Latent Structural Information". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics. 4306–4319. DOI: 10.18653/v1/2020.acl-main.397. URL: https://www.aclweb.org/anthology/2020.acl-main.397.

Zhou, Q., N. Yang, F. Wei, S. Huang, M. Zhou, and T. Zhao. (2018b). "Neural document summarization by jointly learning to score and select sentences". *arXiv preprint arXiv:1807.02305*.

Zhu, H., Y. Lin, Z. Liu, J. Fu, T.-S. Chua, and M. Sun. (2019a). "Graph Neural Networks with Generated Parameters for Relation Extraction". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics. 1331–1339. DOI: 10.18653/v1/P19-1128. URL: https://www.aclweb.org/anthology/P19-1128.

Zhu, J., J. Li, M. Zhu, L. Qian, M. Zhang, and G. Zhou. (2019b). "Modeling Graph Structure in Transformer for Better AMR-to-Text Generation". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics. 5459–5468. DOI: 10.18653/v1/D19-1548. URL: https://www.aclweb.org/anthology/D19-1548.

Zhu, J., J. Li, M. Zhu, L. Qian, M. Zhang, and G. Zhou. (2019c). "Modeling Graph Structure in Transformer for Better AMR-to-Text Generation". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. 5458–5467.

Zhu, Q., Z. Feng, and X. Li. (2018). "GraphBTM: Graph Enhanced Autoencoded Variational Inference for Biterm Topic Model". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics. 4663–4672. DOI: 10.18653/v1/D18-1495. URL: https://www.aclweb.org/anthology/D18-1495 (accessed on 12/26/2020).