Graph4NLP

Deep Learning on Graphs for Natural Language Processing

Yu Chen
Facebook AI

July 8th, 2021

Joint work with Lingfei Wu, Heng Ji, Yunyao Li and Bang Liu

Outline

DLG4NLP Introduction

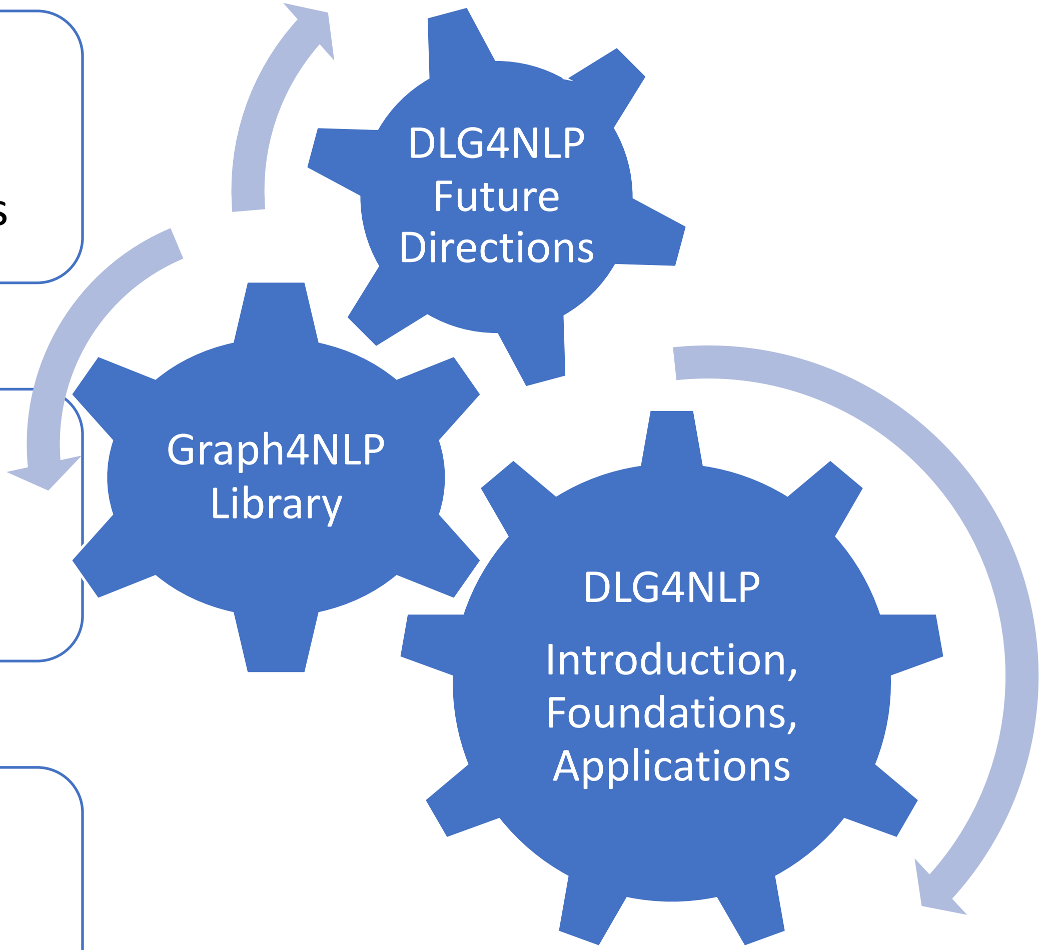
- Why Graphs for NLP?
- Conventional ML for NLP
- Deep Learning on Graphs: Foundations and Models

DLG4NLP Foundations

- Graph Construction for NLP
- Graph Representation Learning for NLP
- Graph Encoder-Decoder Models for NLP

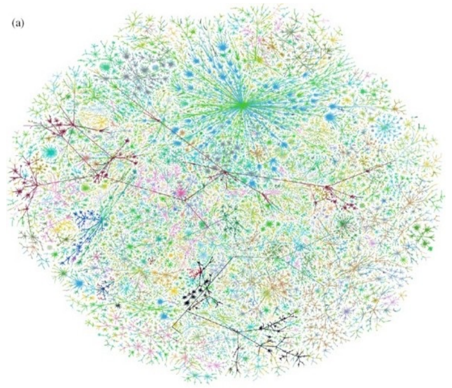
DLG4NLP Applications

- Natural Question Generation
- Summarization

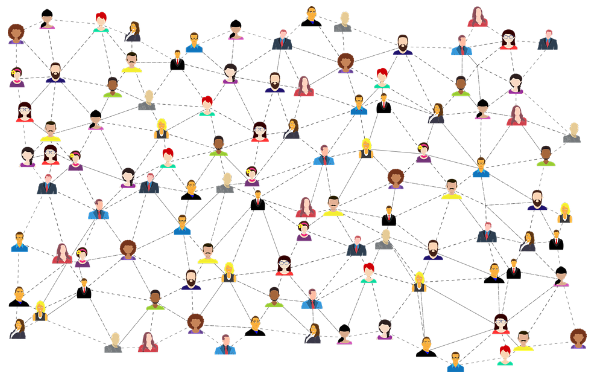


DLG4NLP Introduction

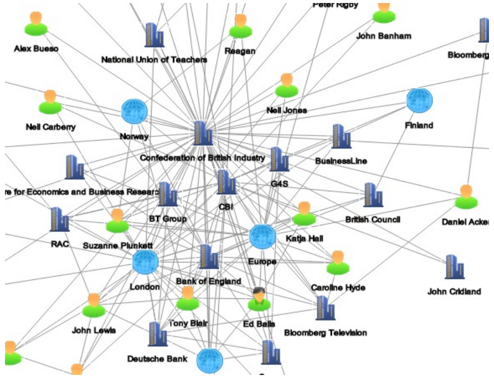
Graph-structured data are ubiquitous



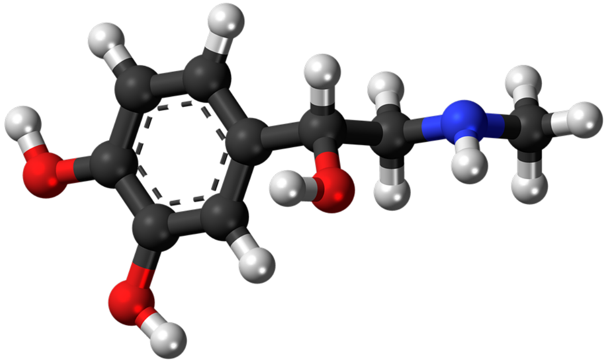
Internet



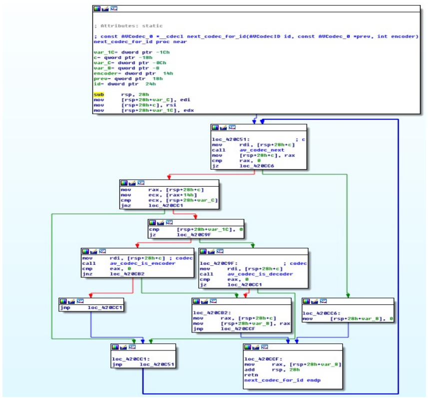
Social networks



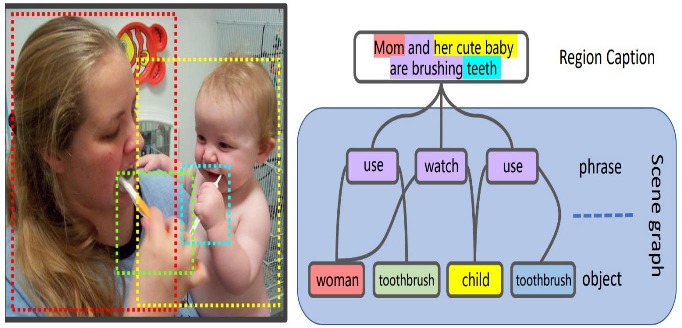
Financial transactions



Biomedical graphs

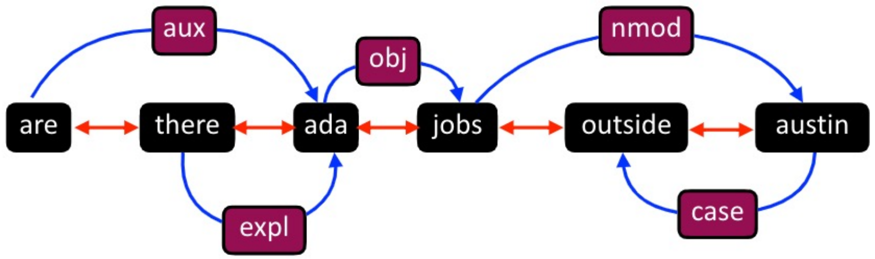


Program graphs

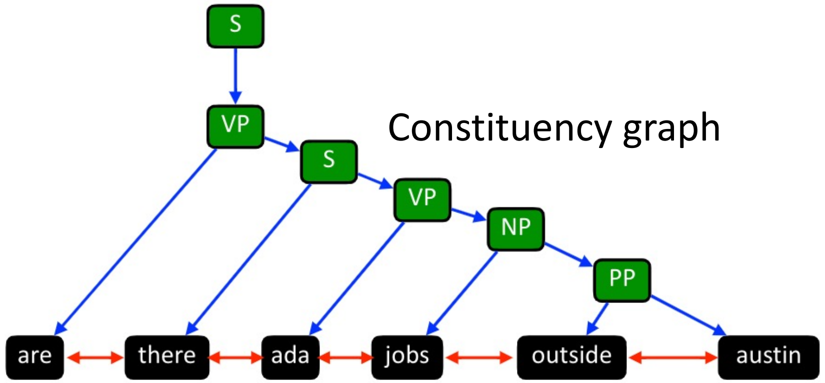


Scene graphs

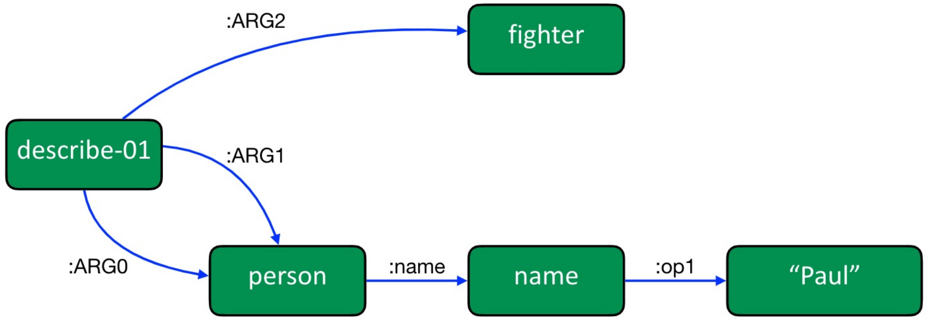
Graphs are ubiquitous in NLP As Well



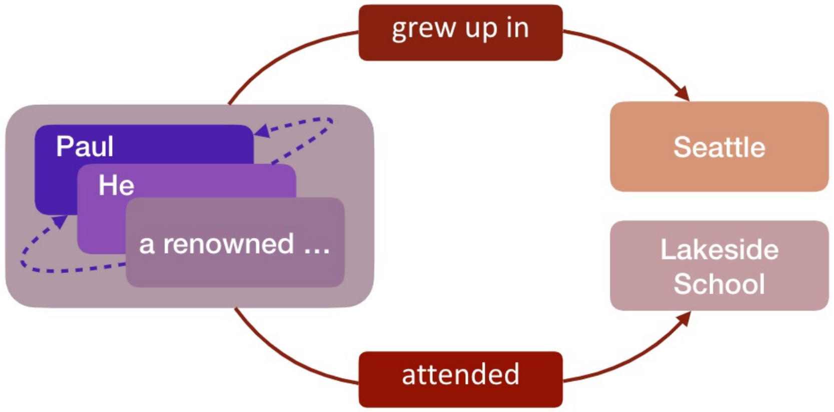
Dependency graph



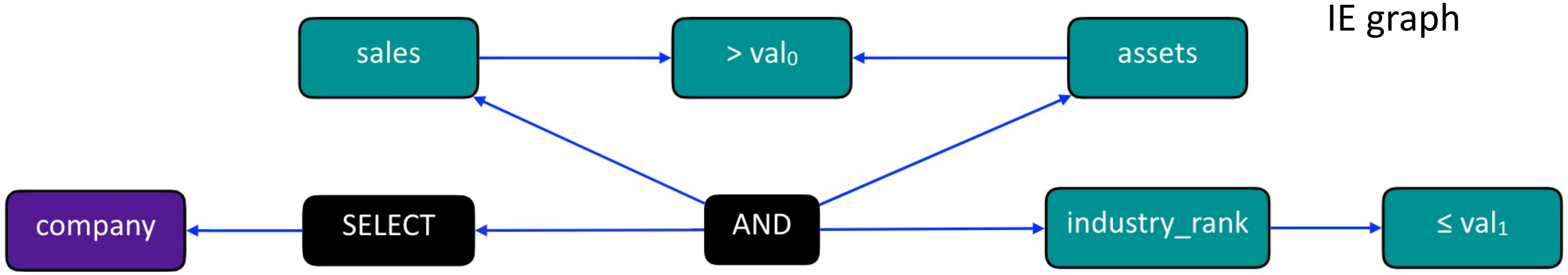
Constituency graph



AMR graph



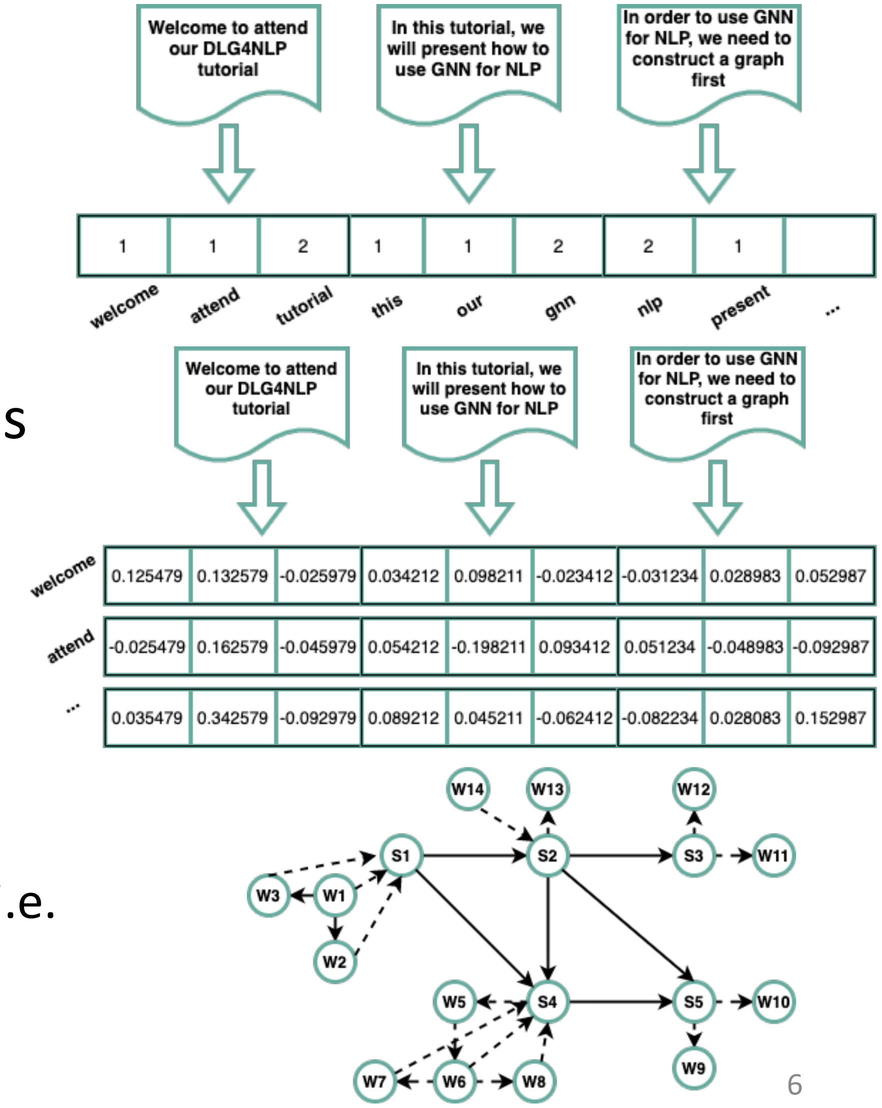
IE graph



SQL graph

Natural Language Processing: A Graph Perspective

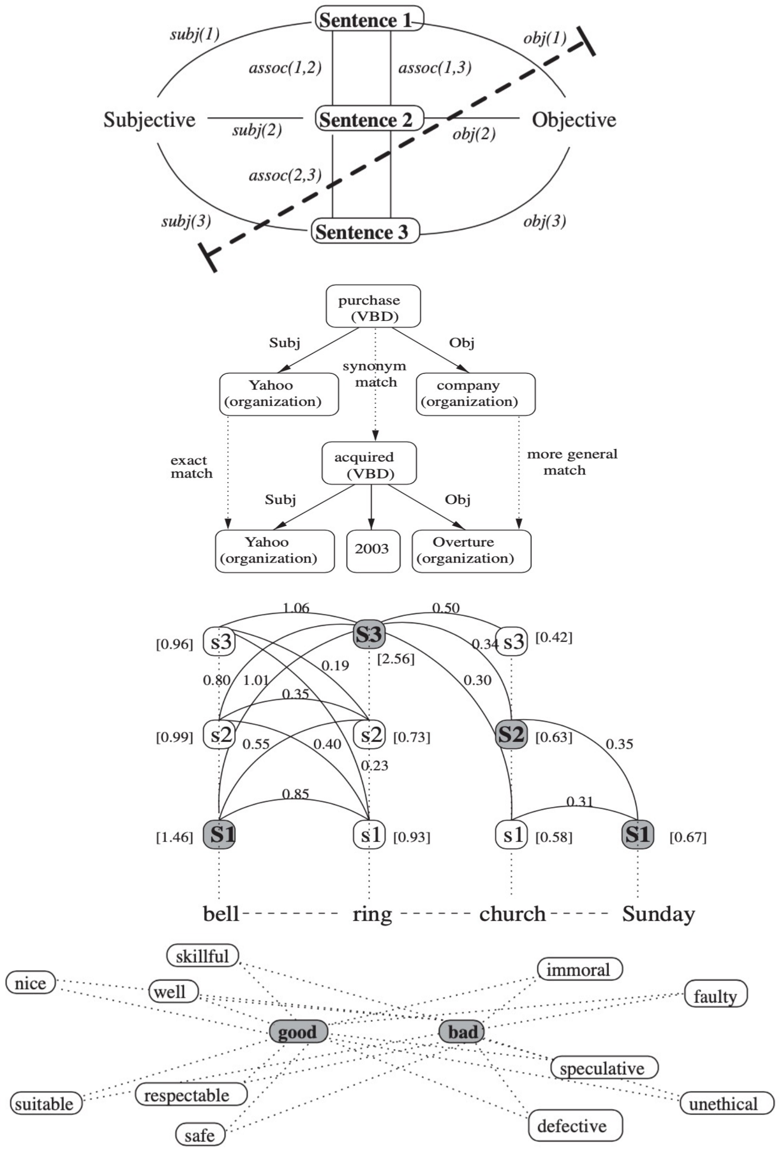
- Represent natural language as a bag of tokens
  - BOW, TF-IDF
  - Topic Modeling: text as a mixture of topics
- Represent natural language as a sequence of tokens
  - Linear-chain CRF
  - Word2vec, Glove
- Represent natural language as a graph
  - Dependency graphs, constituency graphs, AMR graphs, IE graphs, and knowledge graphs
  - Text graph containing multiple hierarchies of elements, i.e. document, sentence and word



Graph Based Methods for NLP

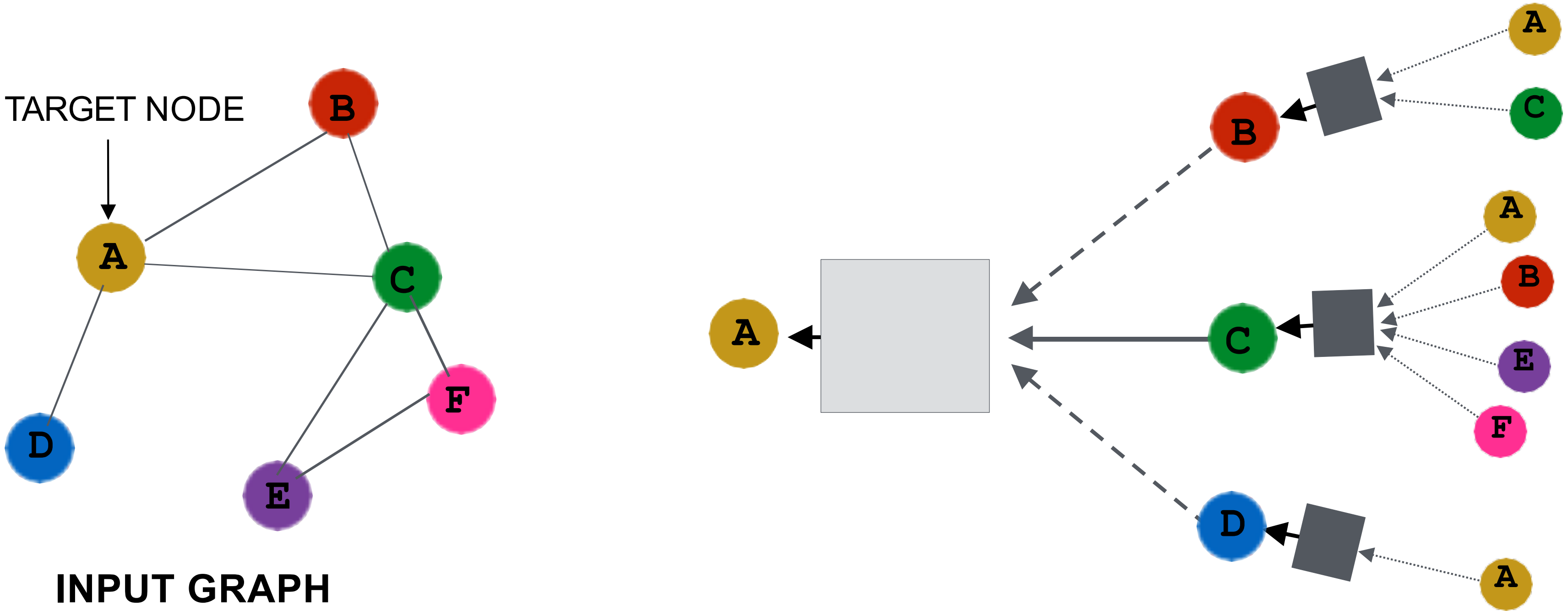
- Random Walk Algorithms
  - Generate random paths, one can obtain a stationary distribution over all the nodes in a graph
  - Applications: semantic similarity of texts, name disambiguation
- Graph Clustering Algorithms
  - Spectral clustering, random walk clustering and min-cut clustering for text clustering
- Graph Matching Algorithms
  - Compute the similarity between two graphs for textual entailment task
- Label Propagation Algorithms
  - Propagate labels from labeled data points to previously unlabeled data points
  - Applications: word-sense disambiguation, sentiment analysis

[Mihalcea and Radev, 2011]



Graph Neural Networks: Basic Model

- Key idea: Generate node embeddings based on local neighborhoods.



Graph4NLP

Graph Neural Networks: Foundations

- Learning node embeddings:

A graph filter

adjacency matrix

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

Updated node embeddings

Input node embeddings

$f_{\text{filter}}(\cdot, \cdot)$
- Spectral-based
- Spatial-based
- Attention-based
- Recurrent-based

- Learning graph-level embeddings:

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

A small graph w/ fewer nodes

New node embeddings

Input graph

Input node embeddings

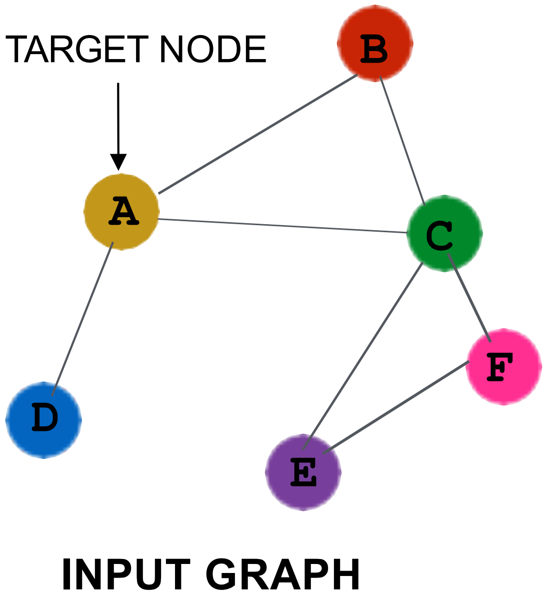
$f_{\text{pool}}(\cdot, \cdot)$
- Flat Graph Pooling (i.e. Max, Ave, Min)
- Hierarchical Graph Pooling (i.e. Diffpool)

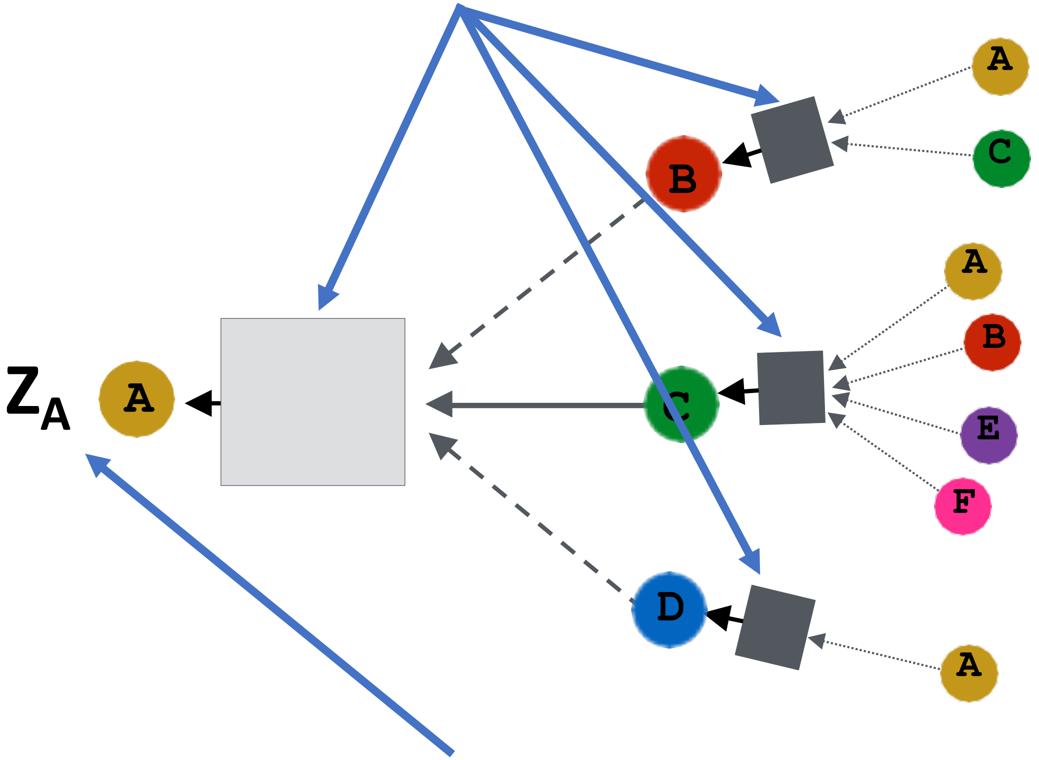
Graph Neural Networks: Popular Models

- Spectral-based Graph Filters
  - GCN (Kipf & Welling, ICLR 2017), Chebyshev-GNN (Defferrard et al. NIPS 2016)
- Spatial-based Graph Filters
  - MPNN (Gilmer et al. ICML 2017), GraphSage (Hamilton et al. NIPS 2017)
  - GIN (Xu et al. ICLR 2019)
- Attention-based Graph Filters
  - GAT (Velickovic et al. ICLR 2018)
- Recurrent-based Graph Filters
  - GGNN (Li et al. ICLR 2016)

Overview of GNN Model

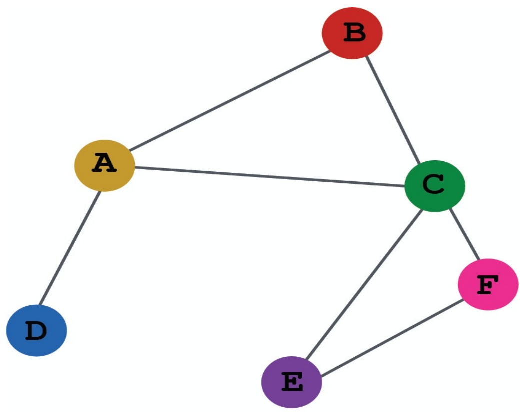
1) Define a neighborhood aggregation function





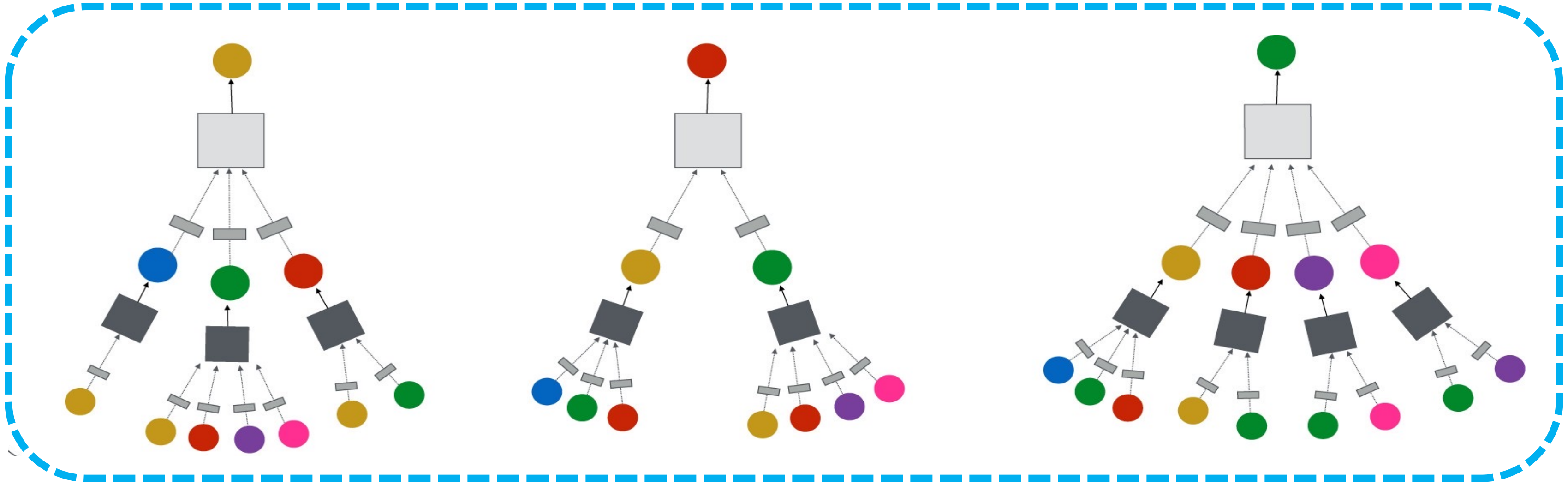
2) Define a loss function on the embeddings, **L(z_v)**

Overview of GNN Model

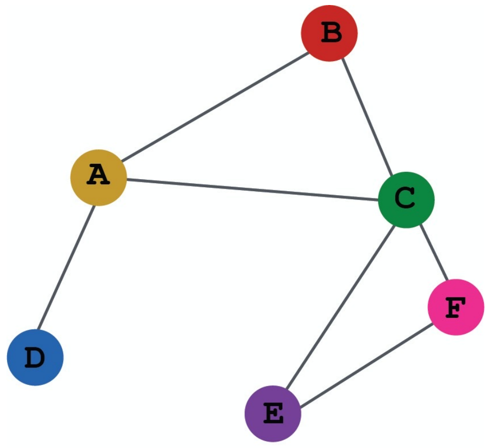


INPUT GRAPH

3) Train on a set of nodes, i.e., a batch of computation graphs



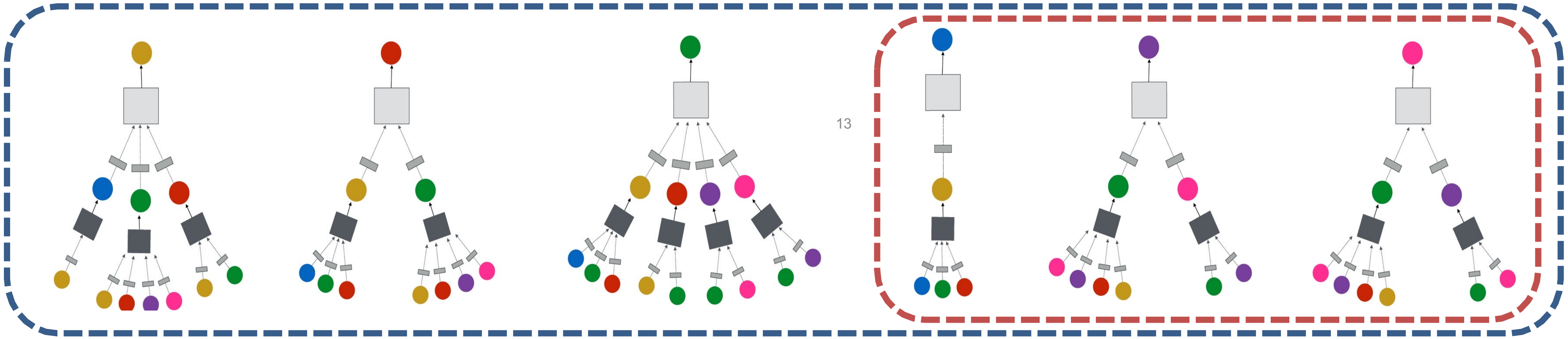
Overview of GNN Model



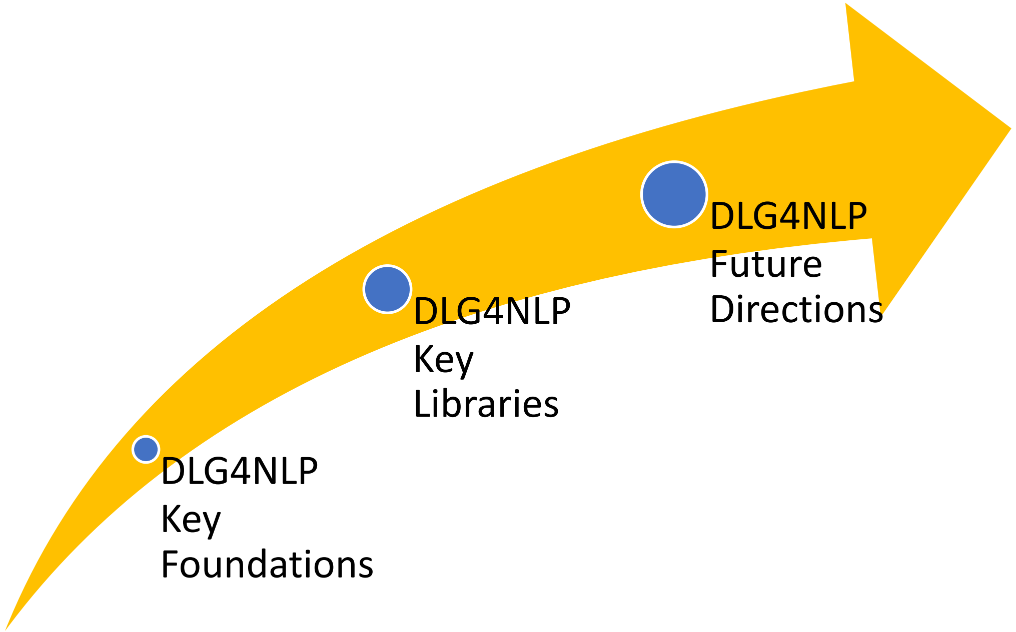
INPUT GRAPH

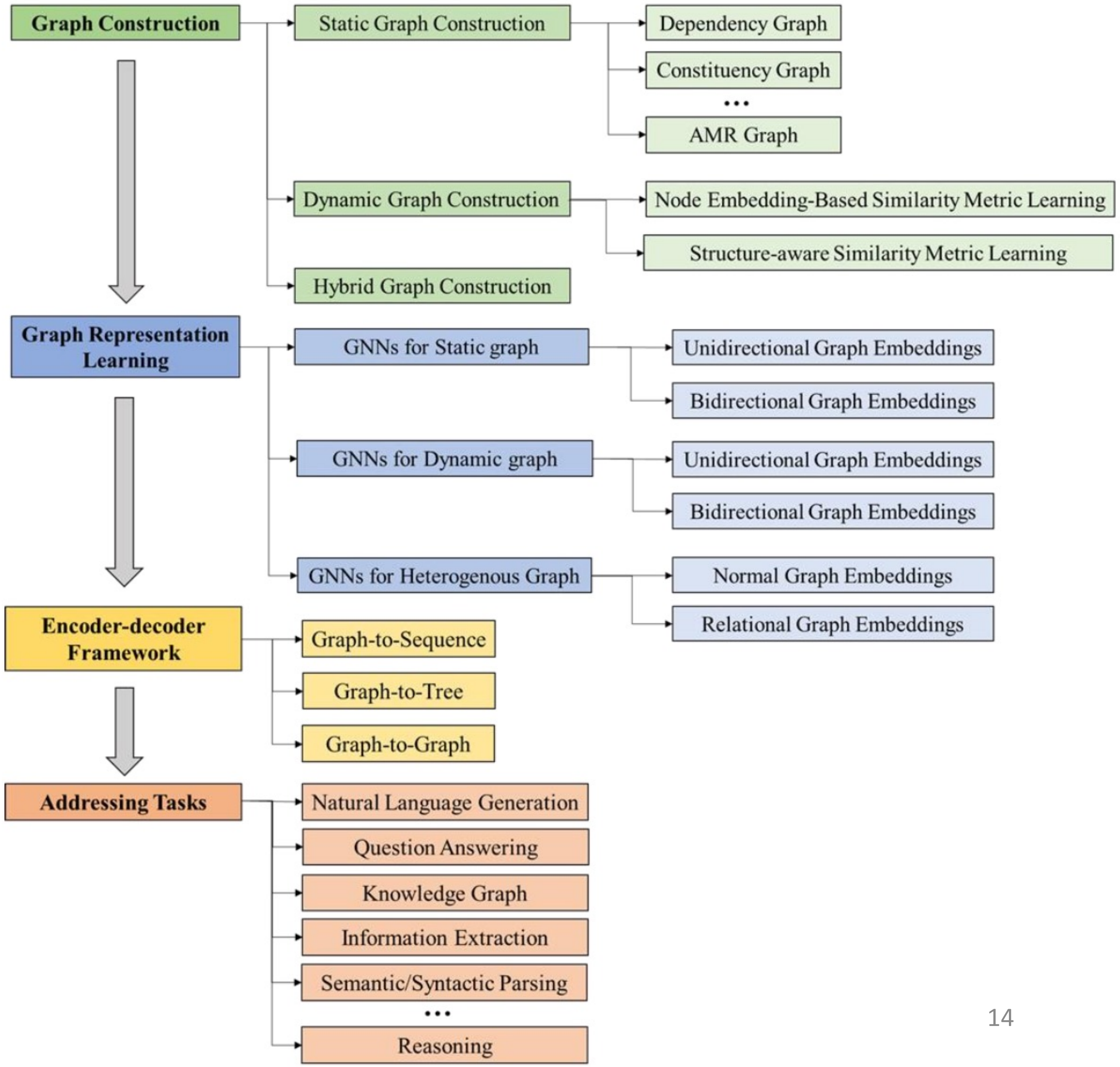
4) Generate embeddings for nodes as needed

Even for nodes we never trained on!



DLG4NLP: A Roadmap





Outline

DLG4NLP Introduction

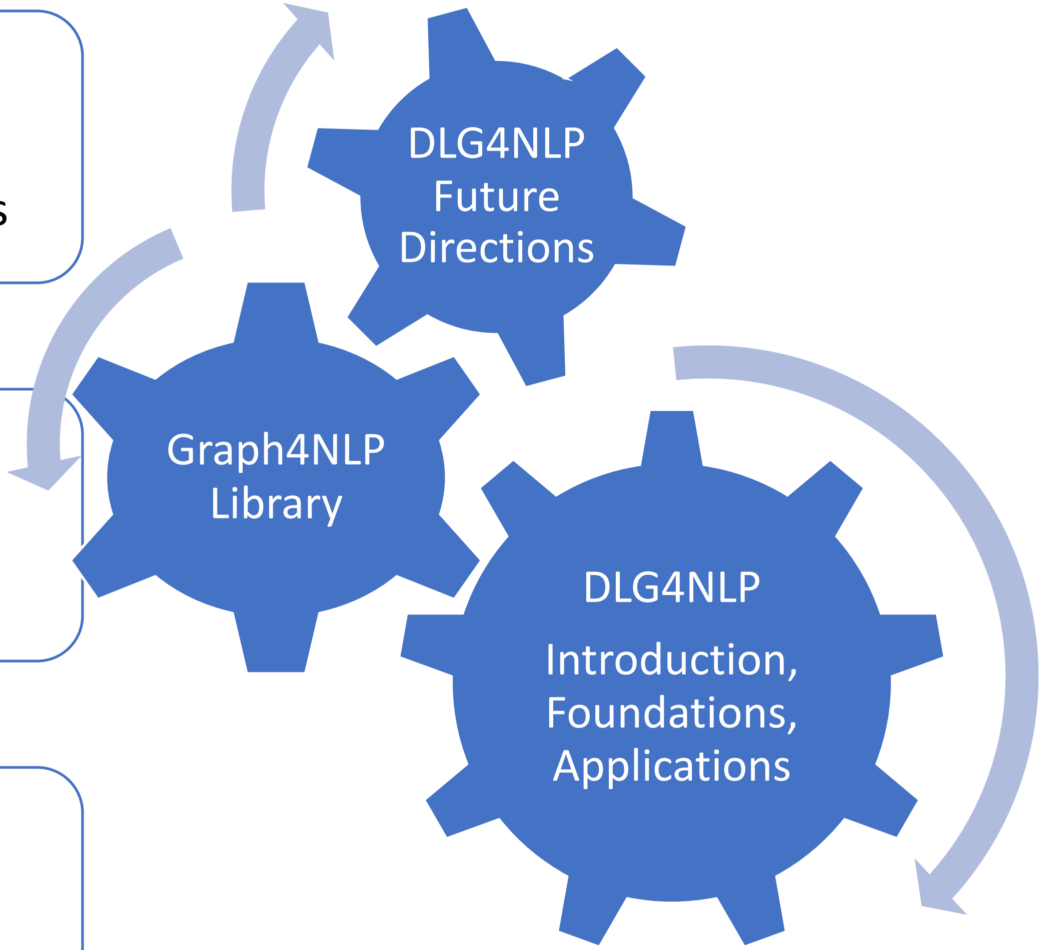
- Why Graphs for NLP?
- Conventional ML for NLP
- Deep Learning on Graphs: Foundations and Models

DLG4NLP Foundations

- Graph Construction for NLP
- Graph Representation Learning for NLP
- Graph Encoder-Decoder Models for NLP

DLG4NLP Applications

- Natural Question Generation
- Summarization



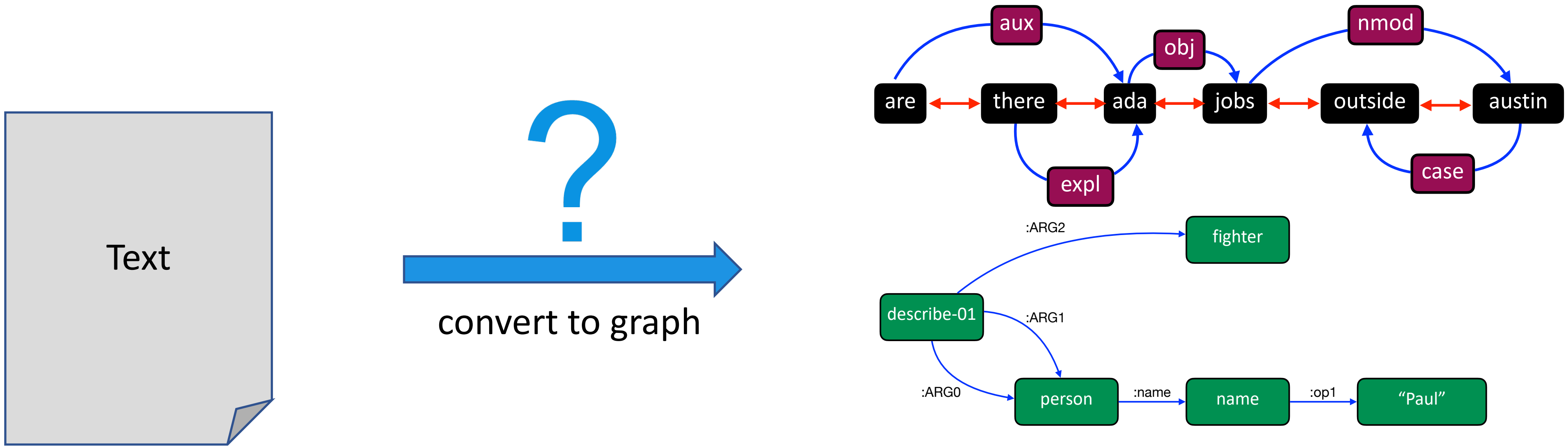
Graph4NLP

DLG4NLP
Foundations

Graph Construction for NLP

Why Graph Construction for NLP?

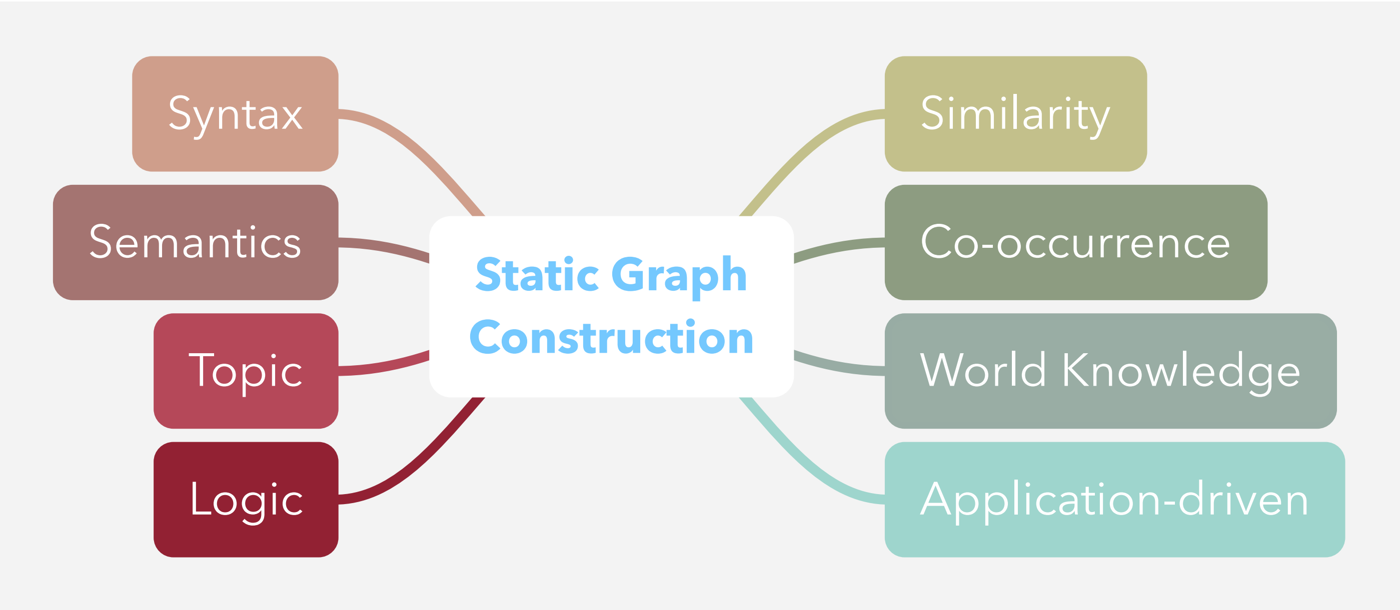
- Representation power: graph > sequence > bag
- Different NLP tasks require different aspects of text , e.g., syntax, semantics.
- Different graphs capture different aspects of the text
- Two categories: static vs dynamic graph construction
- Goal: good downstream task performance



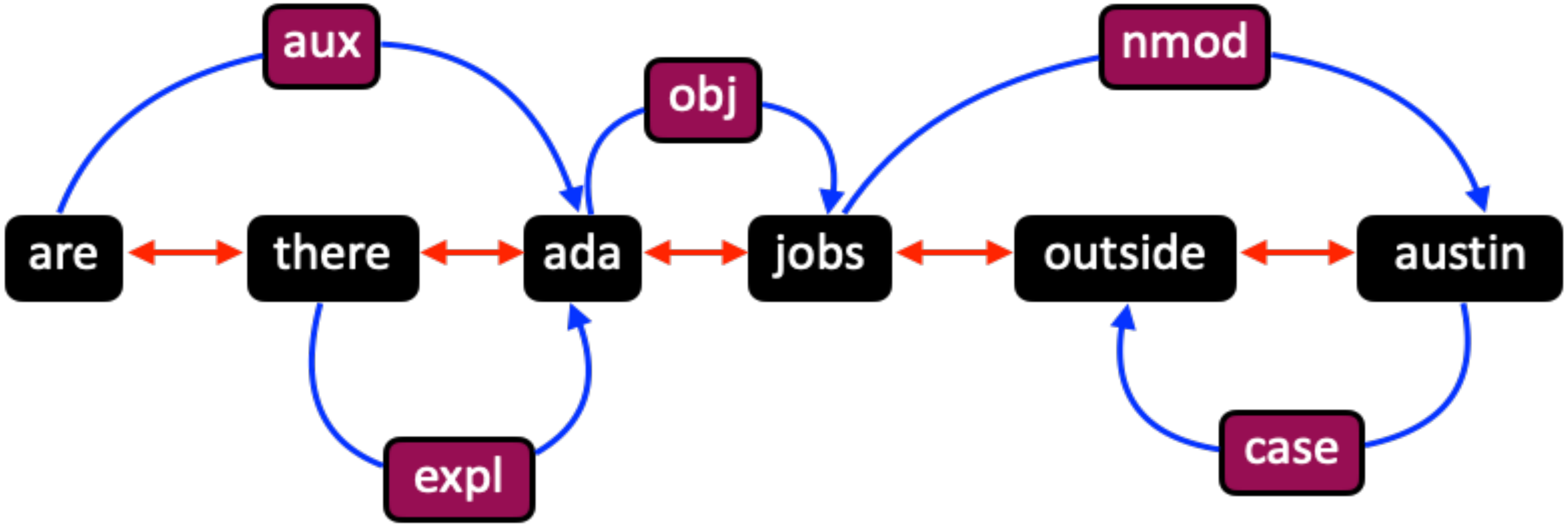
many more graph options…

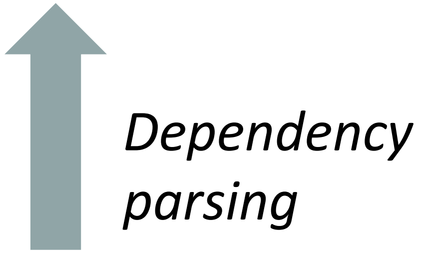
Static Graph Construction

- Problem setting:
  - Input: raw text (e.g., sentence, paragraph, document, corpus)
  - Output: graph
- Conducted during preprocessing by augmenting text with domain knowledge



Static Graph Construction: Dependency Graph

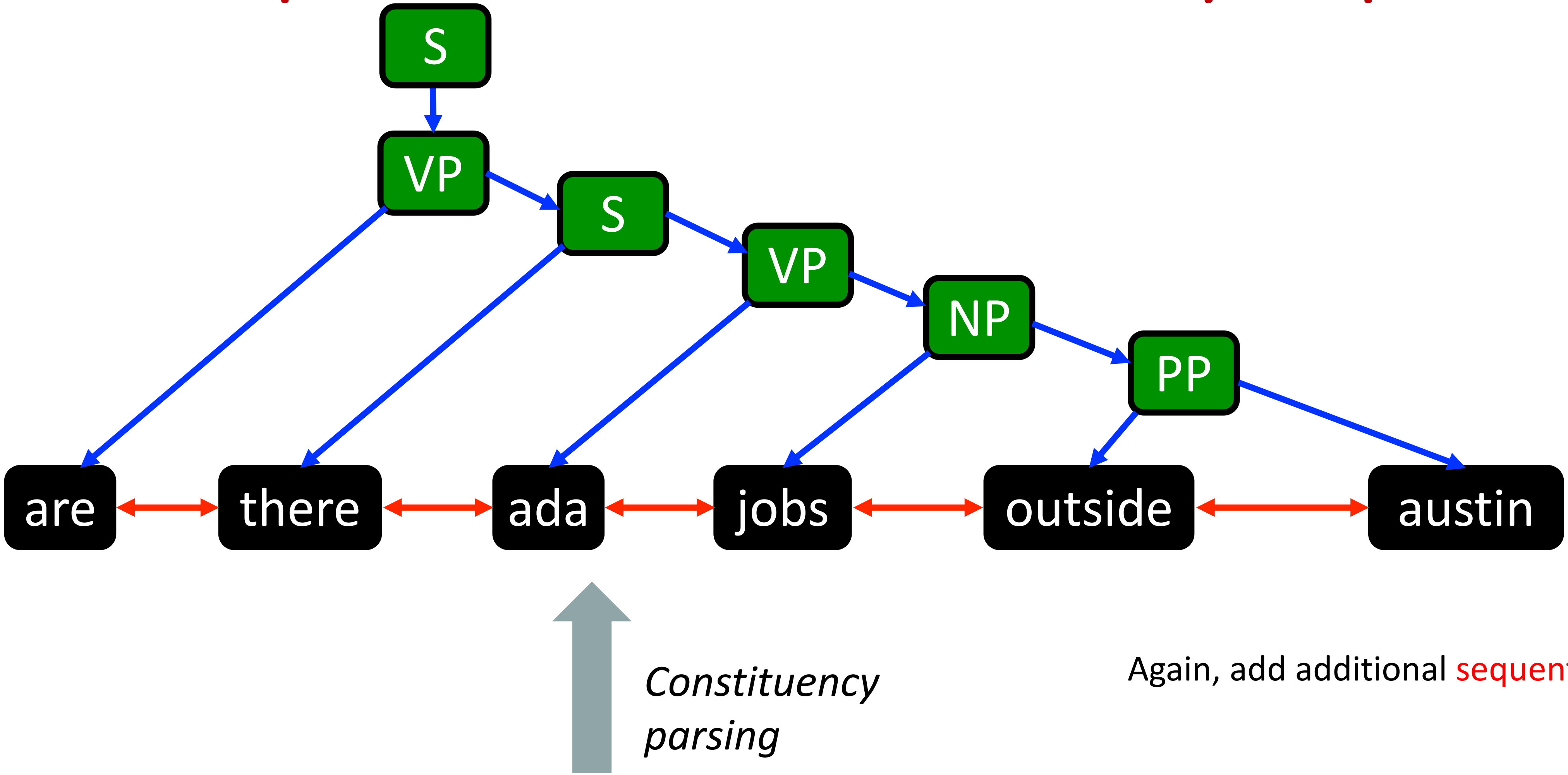




Text input: are there ada jobs outside austin

Add additional sequential edges to
1) reserve sequential information in raw text
2) connect multiple dependency graphs in a paragraph

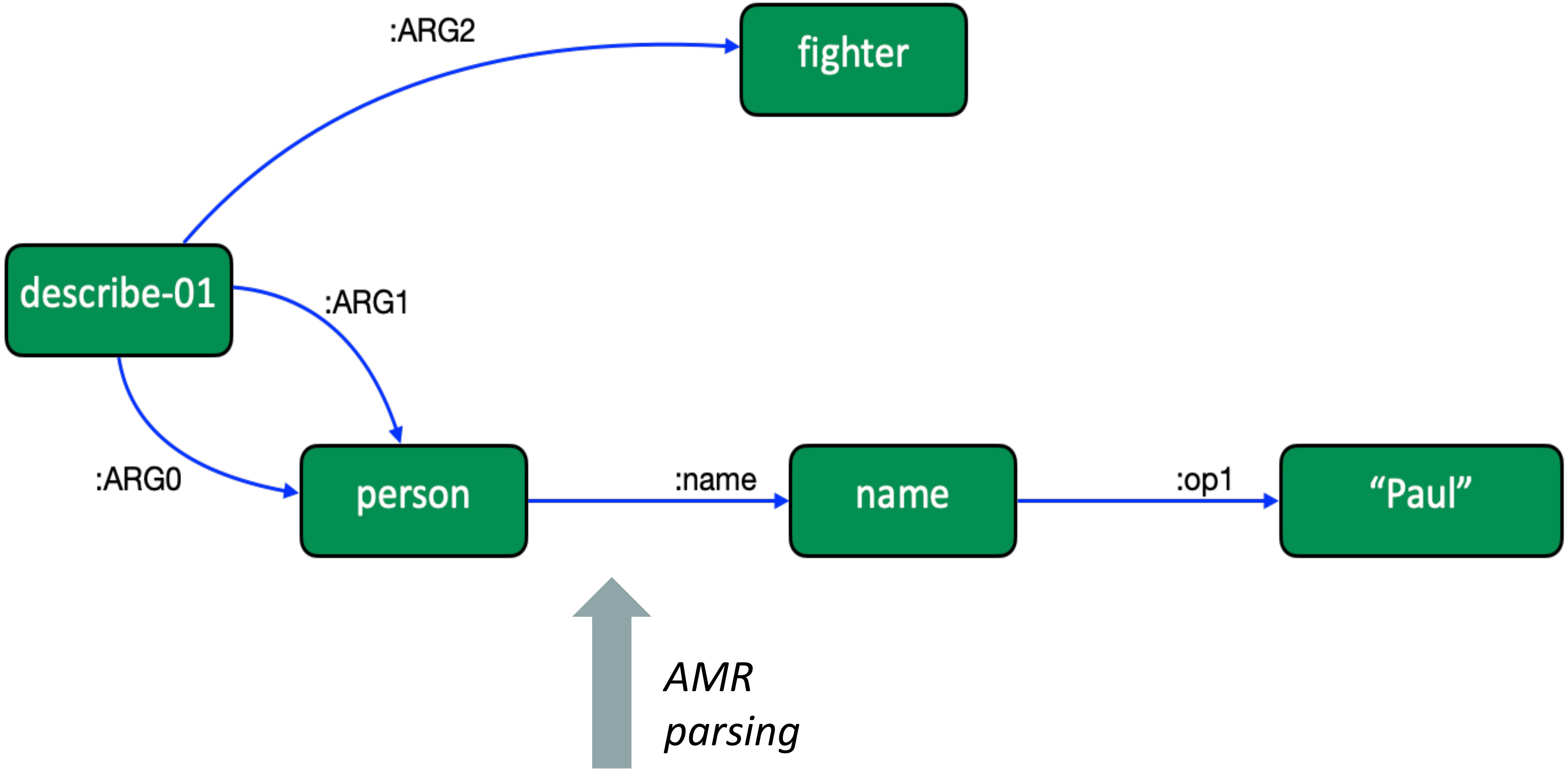
Static Graph Construction: Constituency Graph



Again, add additional sequential edges

Text input: are there ada jobs outside austin

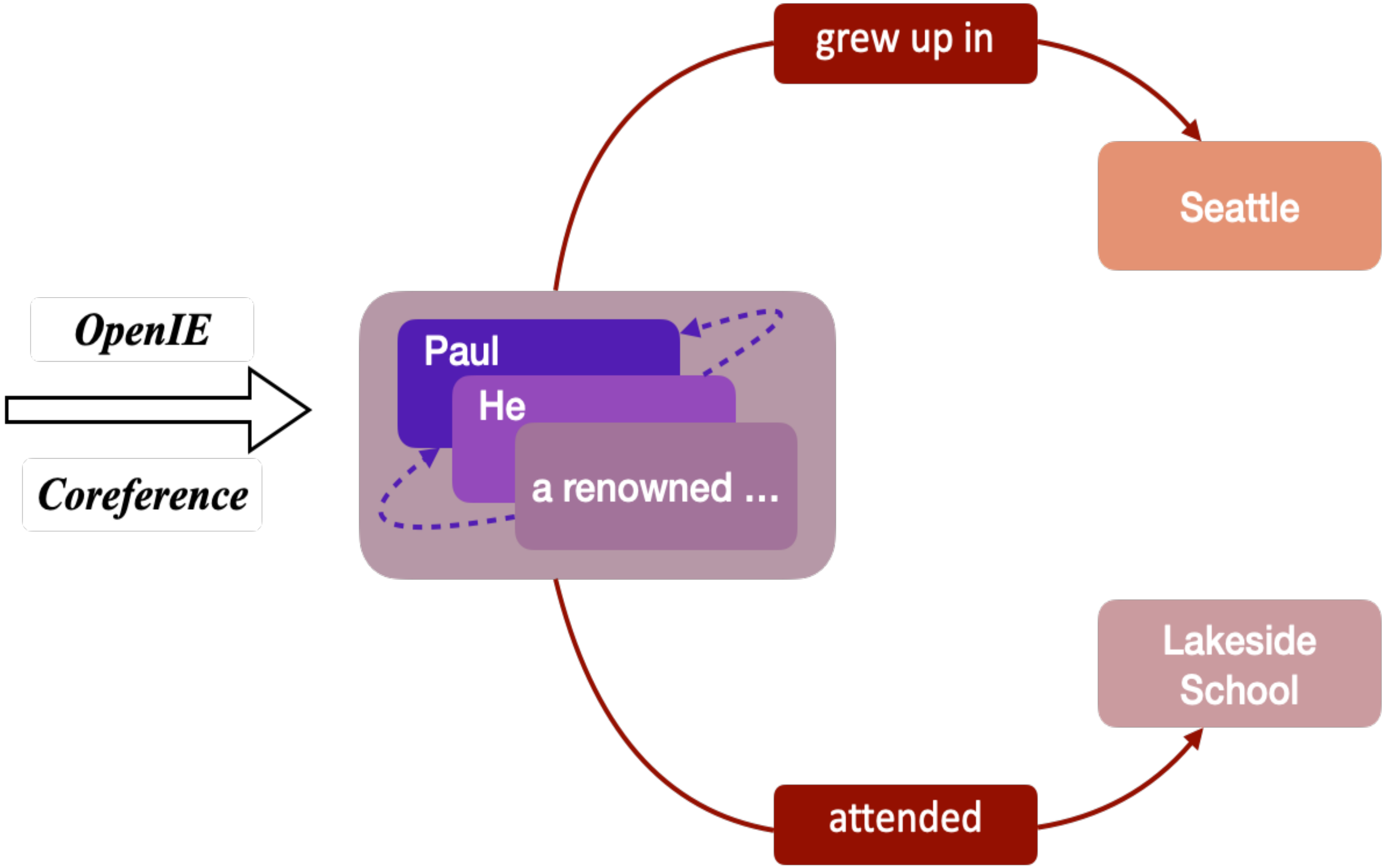
Static Graph Construction: AMR Graph



Text input: Paul’s description of himself: a fighter

Static Graph Construction: IE Graph

***Text input**: Paul, a renowned computer scientist, grew up in Seattle. He attended Lakeside School.*



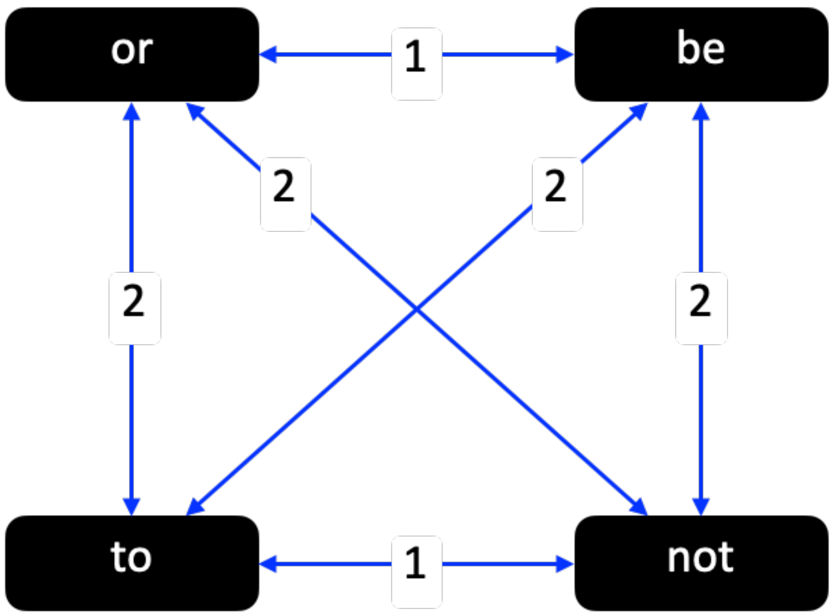
Static Graph Construction: Co-occurrence Graph

***Text input**: To be, or not to be: ...*

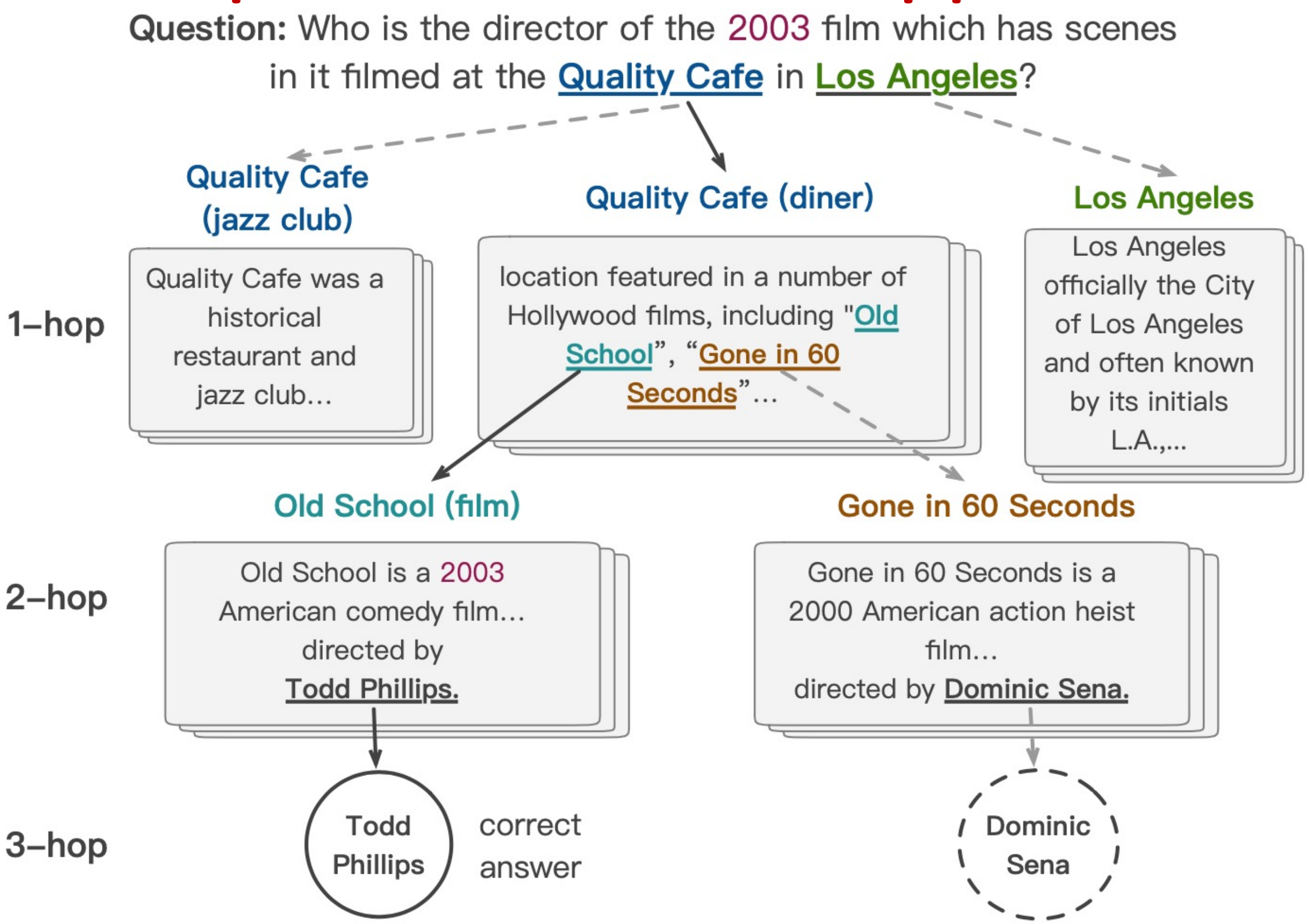
Co-occurrence matrix

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

Co-occurrence graph

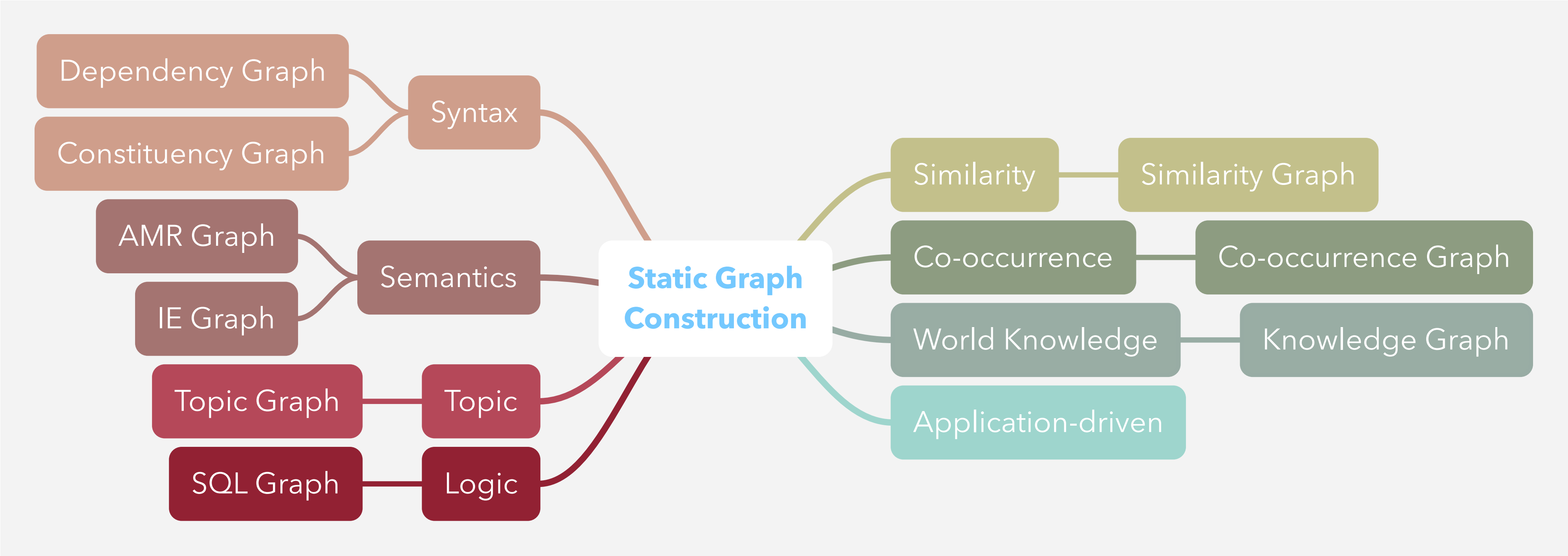


Static Graph Construction: Application-driven Graph



Ming Ding et al. "Cognitive Graph for Multi-Hop Reading Comprehension at Scale". ACL 2019.

Static Graph Construction: Summary

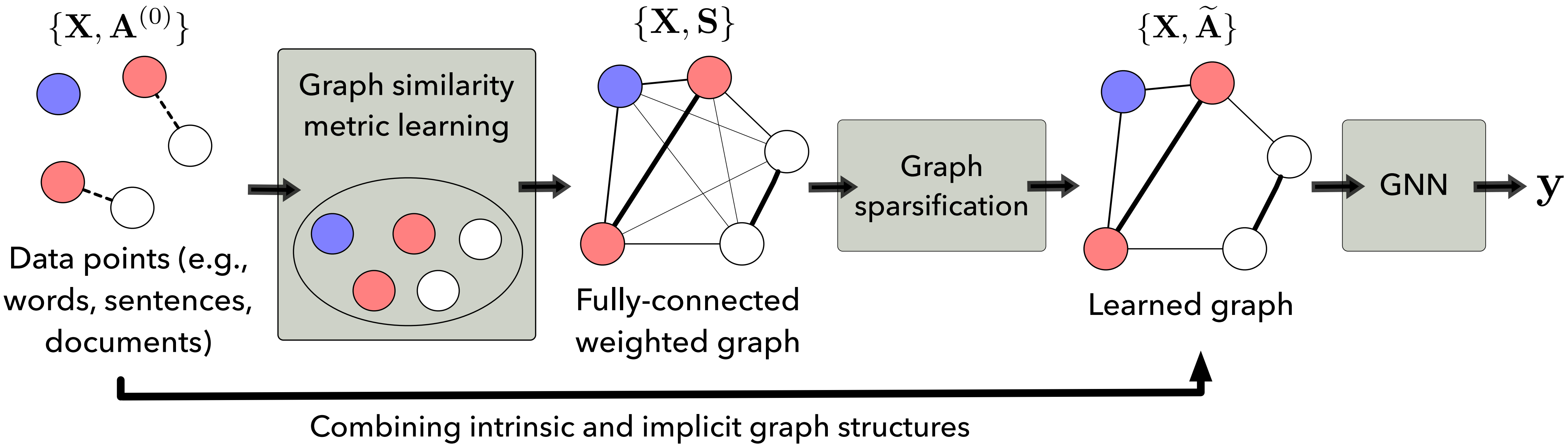


Widely used in various NLP applications such as NLG, MRC, semantic parsing, etc.

Dynamic Graph Construction

- Problem setting:
  - Input: raw text (e.g., sentence, paragraph, document, corpus)
  - Output: graph
- Graph structure (adjacency matrix) learning on the fly, joint with graph representation learning

Dynamic Graph Construction: Overview

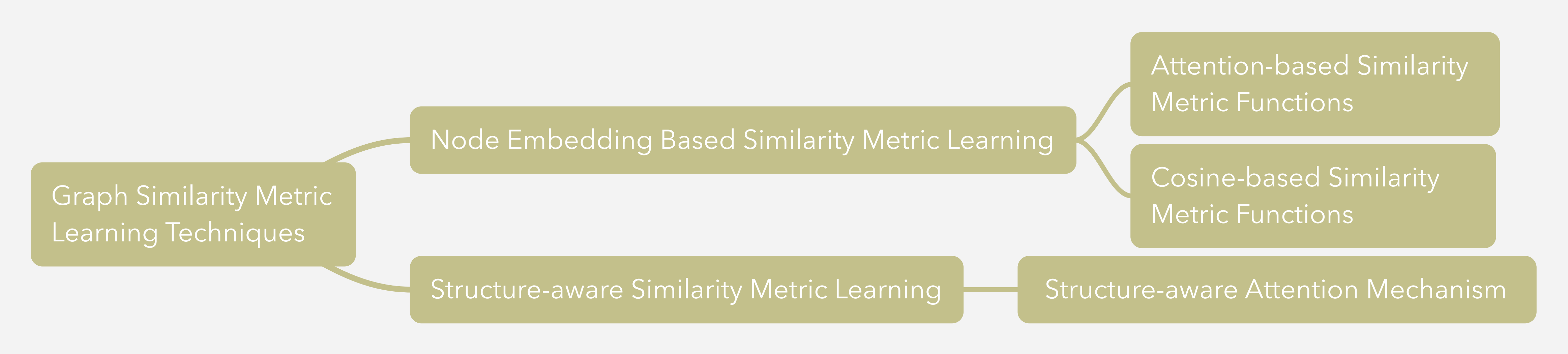


Dynamic Graph Construction Outline

- **Dynamic Graph Construction**
  - Graph Similarity Metric Learning Techniques
  - Graph Sparsification Techniques
  - Combining Intrinsic Graph Structures and Implicit Graph Structures
  - Learning Paradigms

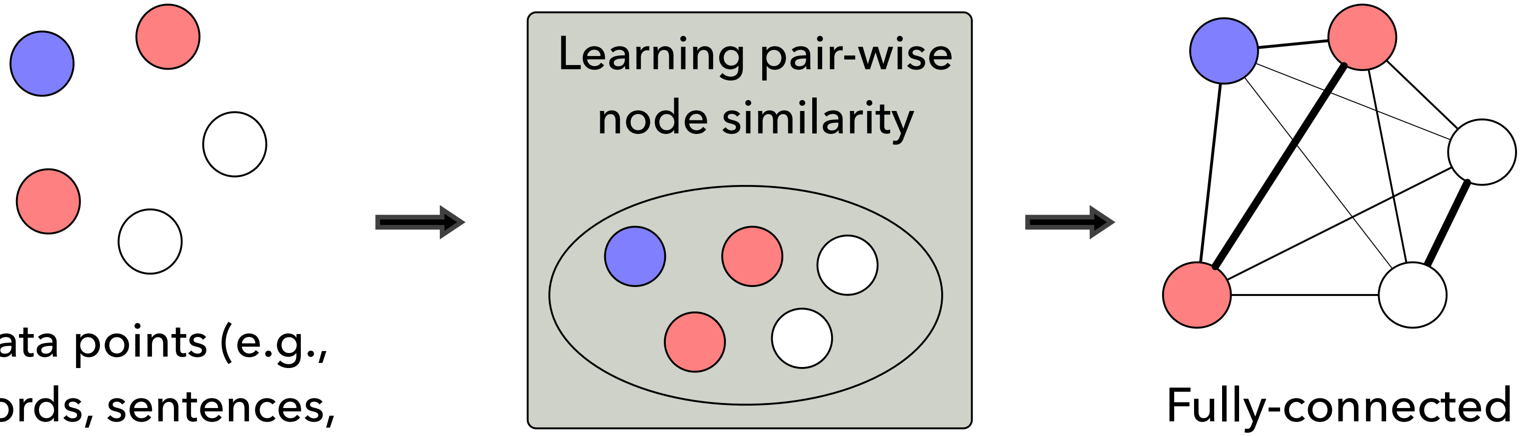
Graph Similarity Metric Learning Techniques

- Graph structure learning as similarity metric learning (in the node embedding space)
- Enabling inductive learning
- Various metric functions



Node Embedding Based Similarity Metric Learning

- Learning a weighted adjacency matrix by computing the pair-wise node similarity in the embedding space
- Common metrics functions
  - Attention-based similarity metric functions
  - Cosine-based similarity metric functions



Data points (e.g., words, sentences, documents)

Fully-connected weighted graph

Attention-based Similarity Metric Functions

Variant 1)

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

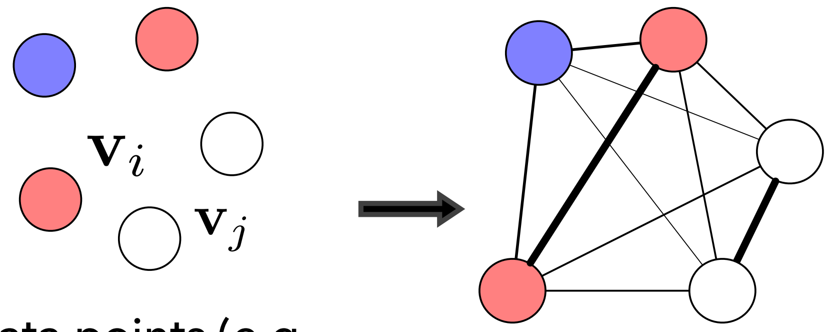
Node feature vector

Non-negative learnable weight vector

Variant 2)

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

Learnable weight matrix



Data points (e.g., words, sentences, documents)

Fully-connected weighted graph

Chen at al. "GraphFlow: Exploiting Conversation Flow with Graph Neural Networks for Conversational Machine Comprehension". IJCAI 2020.

Chen et al. "Reinforcement Learning Based Graph-to-Sequence Model for Natural Question Generation". ICLR 2020.

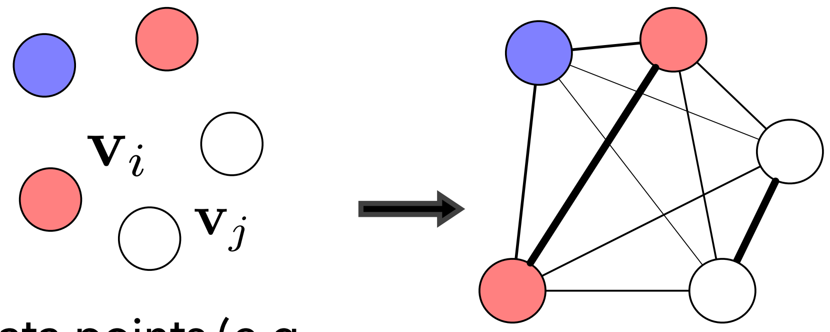
Cosine-based Similarity Metric Functions

$$S_{i,j}^{p} = \cos(\mathbf{w}_p \odot \mathbf{v}_i, \mathbf{w}_p \odot \mathbf{v}_j)$$

Learnable weight vector

$$S_{i,j} = \frac{1}{m}\sum_{p=1}^{m} S_{ij}^{p}$$

Multi-head similarity scores



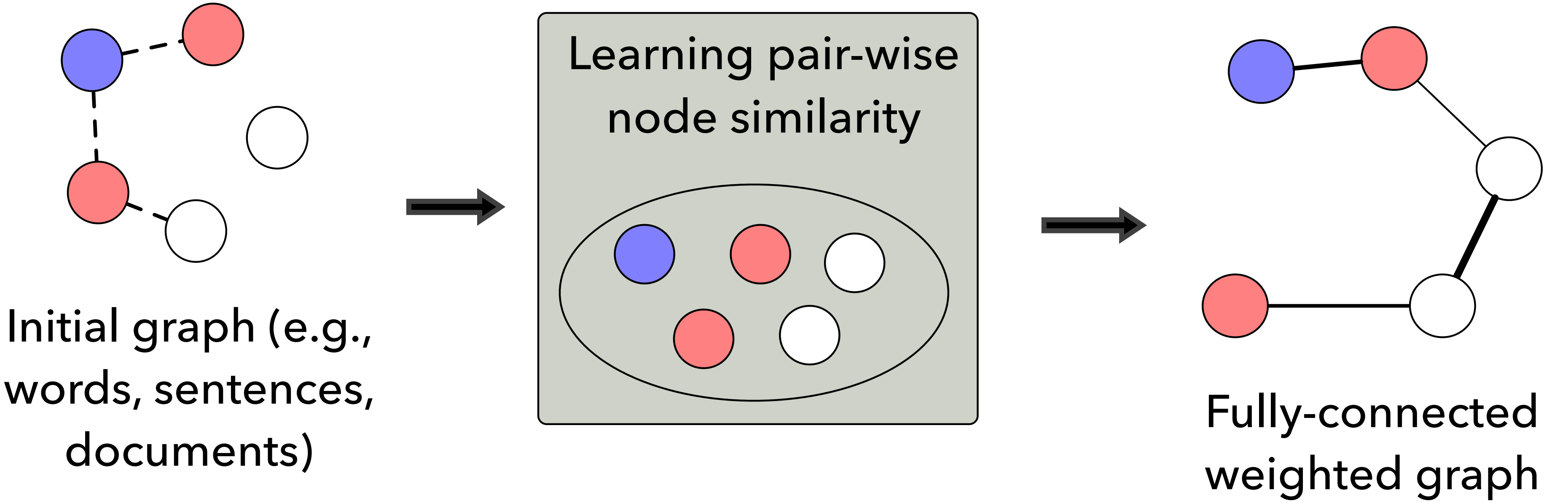
Data points (e.g., words, sentences, documents)

Fully-connected weighted graph

Chen et al. "Iterative Deep Graph Learning for Graph Neural Networks: Better and Robust Node Embeddings". NeurIPS 2021.

Structure-aware Similarity Metric Learning

- Learning a weighted adjacency matrix by computing the pair-wise node similarity in the embedding space
- Considering existing edge information of the intrinsic graph in addition to the node information



Attention-based Similarity Metric Functions

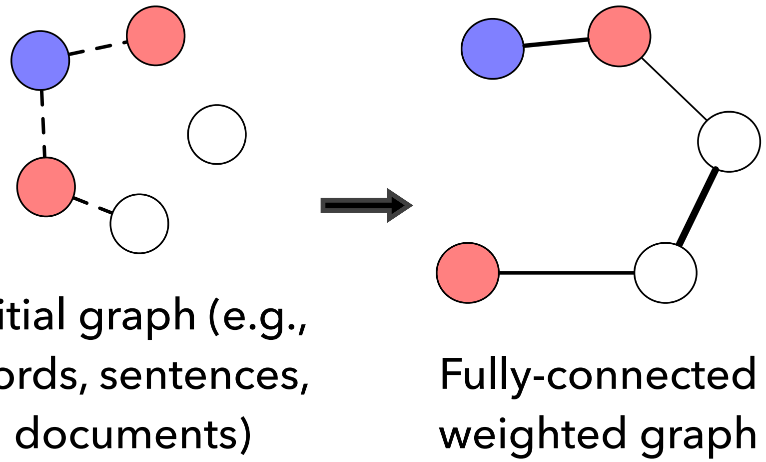
Variant 1)

$$S^l_{i,j} = \mathrm{softmax}(\mathbf{u}^T \tanh(\mathbf{W}[\mathbf{h}^l_i, \mathbf{h}^l_j, \mathbf{v}_i, \mathbf{v}_j, \mathbf{e}_{i,j}]))$$

Edge embeddings

Variant 2)

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

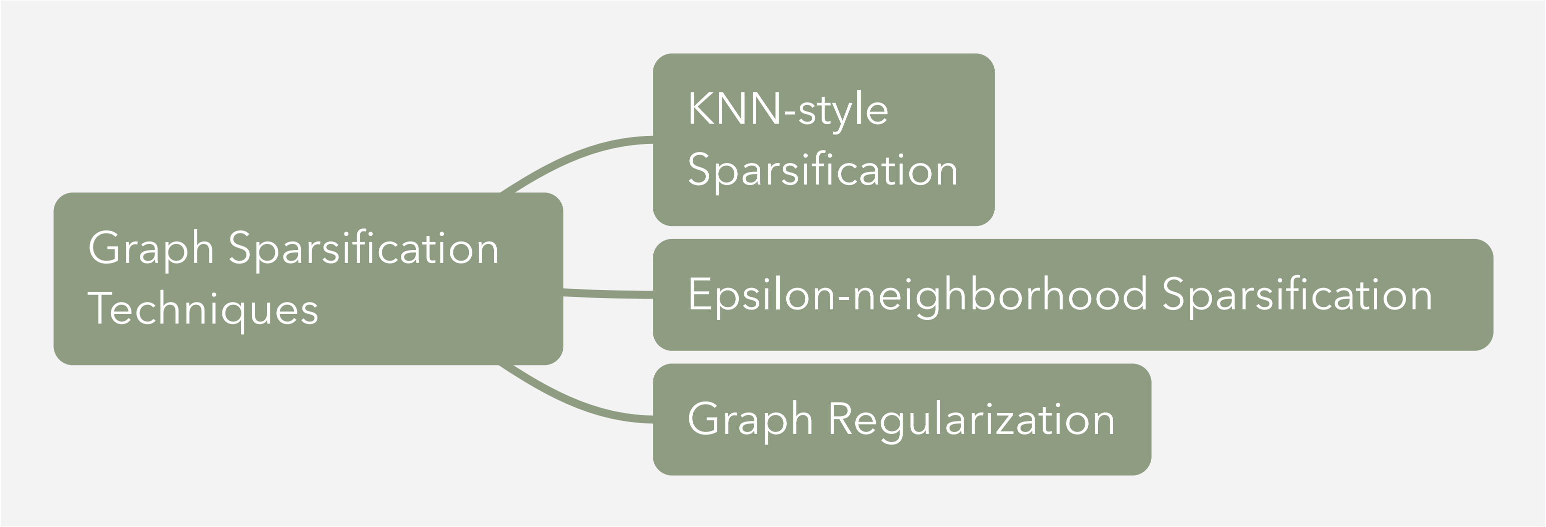


Liu et al. *"Contextualized Non-local Neural Networks for Sequence Learning". AAAI 2019.*

Liu et al. *"Retrieval-Augmented Generation for Code Summarization via Hybrid GNN". ICLR 2021.*

Graph Sparsification Techniques

- Similarity metric functions learn a fully-connected graph
- Fully-connected graph is computationally expensive and might introduce noise
- Enforcing sparsity to the learned graph structure
- Various techniques



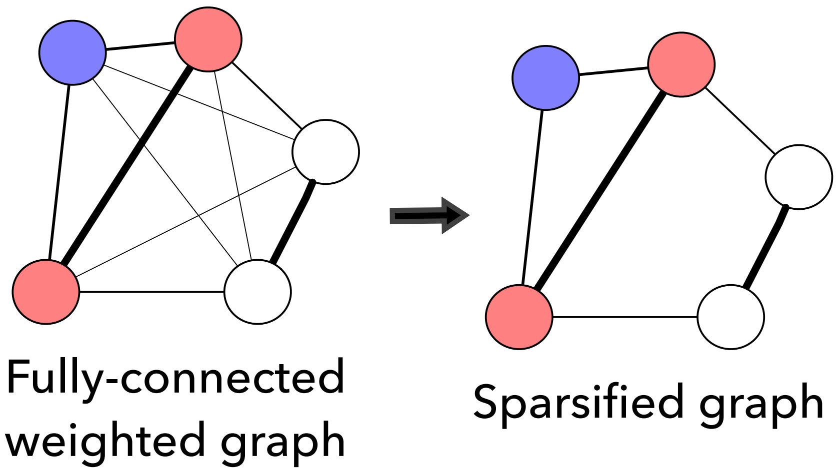
Common Graph Sparsification Options

Option 1) KNN-style Sparsification

$$\mathbf{A}_{i,:} = \mathrm{topk}(\mathbf{S}_{i,:})$$

Option 2) epsilon-neighborhood Sparsification

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



Option 3) graph Regularization

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

Combining Intrinsic and Implicit Graph Structures

- Intrinsic graph typically still carries rich and useful information
- Learned implicit graph is potentially a “shift” (e.g., substructures) from the intrinsic graph structure

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

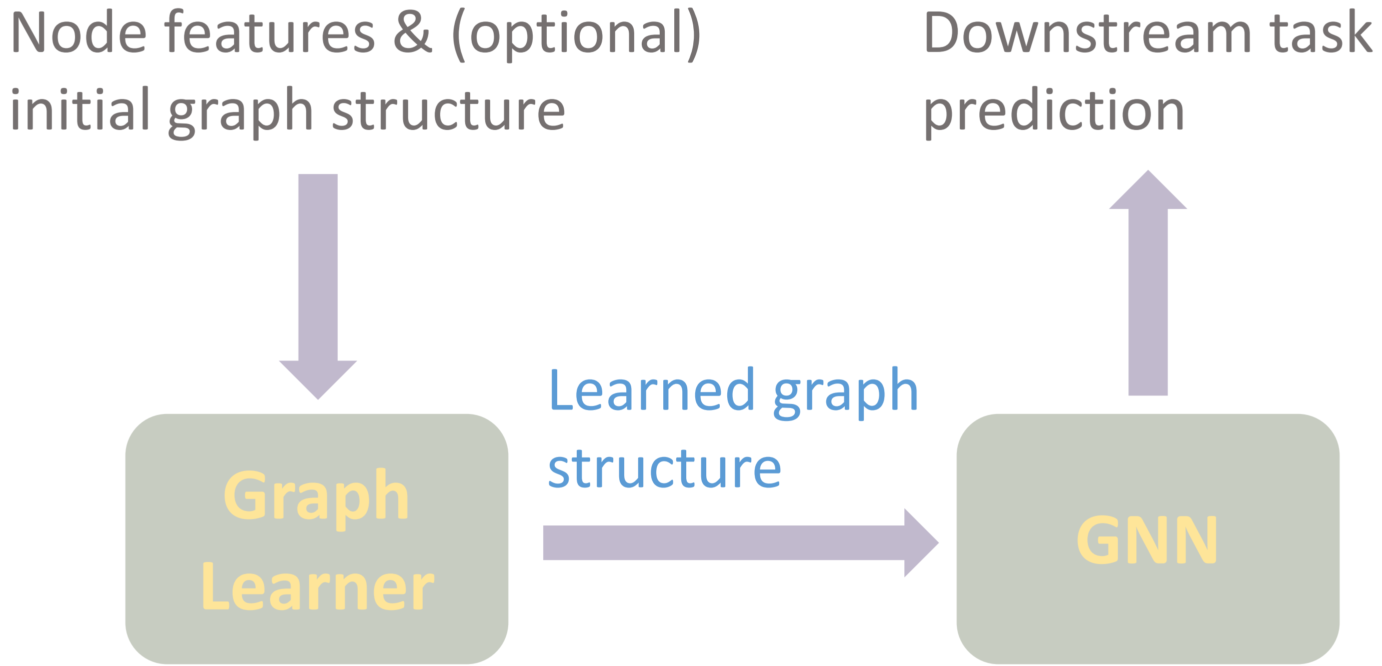
Normalized graph Laplacian

f(A) can be arbitrary operation, e.g., graph Laplacian, row-normalization

Li et al. “Adaptive Graph Convolutional Neural Networks”. AAAI 2018.

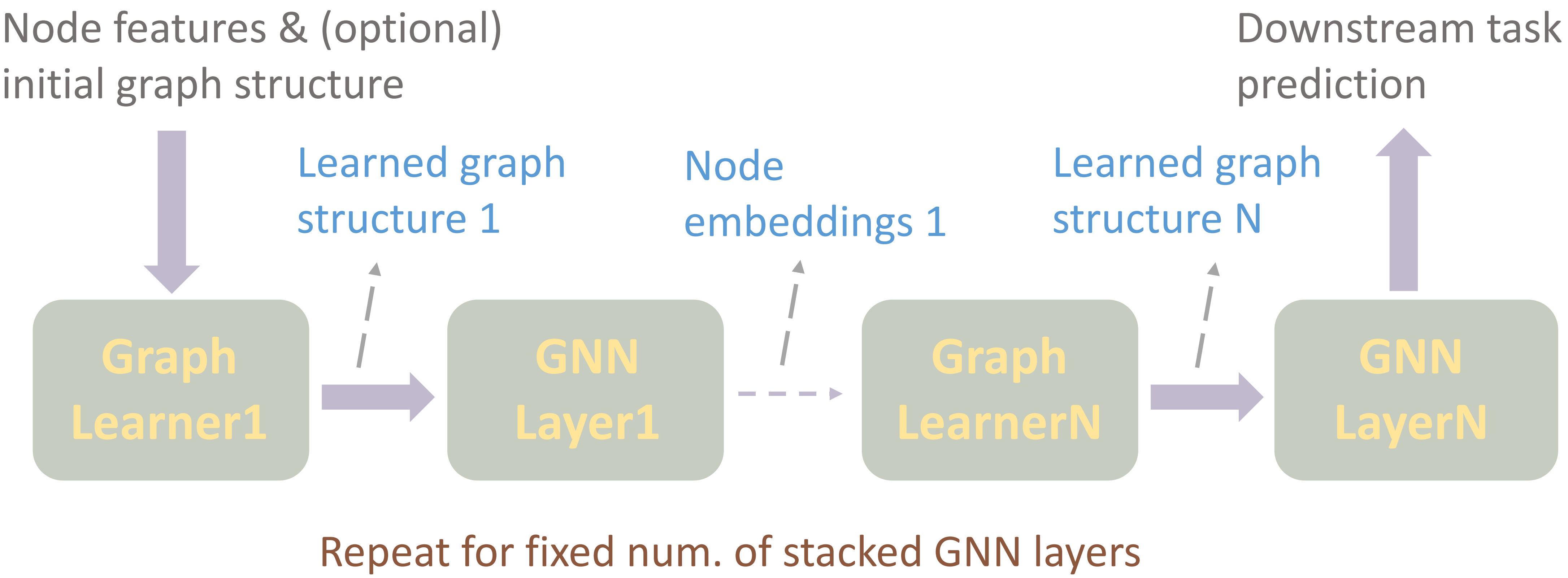
Chen et al. “Iterative Deep Graph Learning for Graph Neural Networks: Better and Robust Node Embeddings”. NeurIPS 2021.

Learning Paradigms: Joint Learning



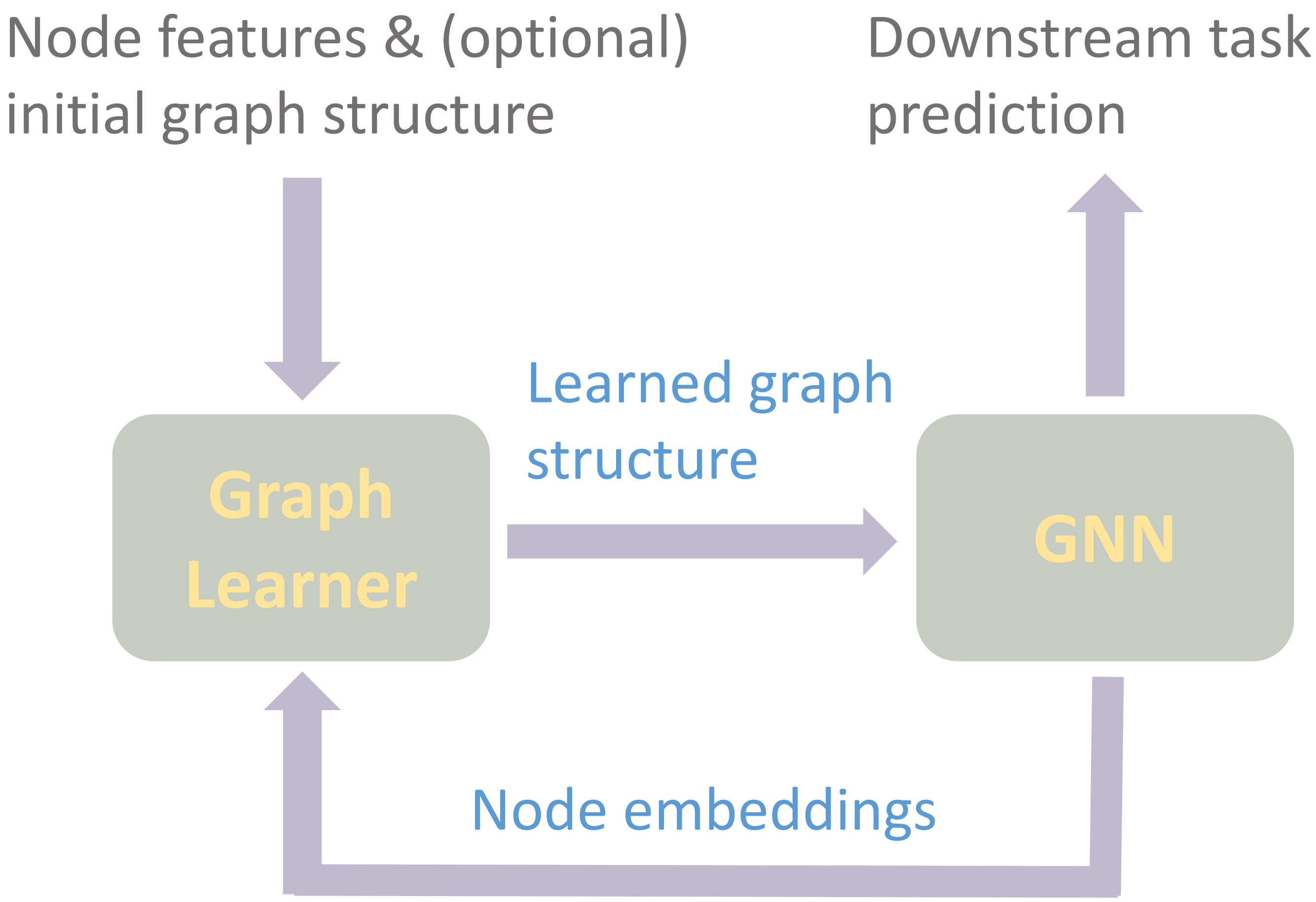
Chen at al. "GraphFlow: Exploiting Conversation Flow with Graph Neural Networks for Conversational Machine Comprehension". IJCAI 2020.
Chen et al. "Reinforcement Learning Based Graph-to-Sequence Model for Natural Question Generation". ICLR 2020.
Liu et al. "Contextualized Non-local Neural Networks for Sequence Learning". AAAI 2019.
Liu et al. "Retrieval-Augmented Generation for Code Summarization via Hybrid GNN". ICLR 2021.

Learning Paradigms: Adaptive Learning



Li et al. “Adaptive Graph Convolutional Neural Networks”. AAAI 2018.

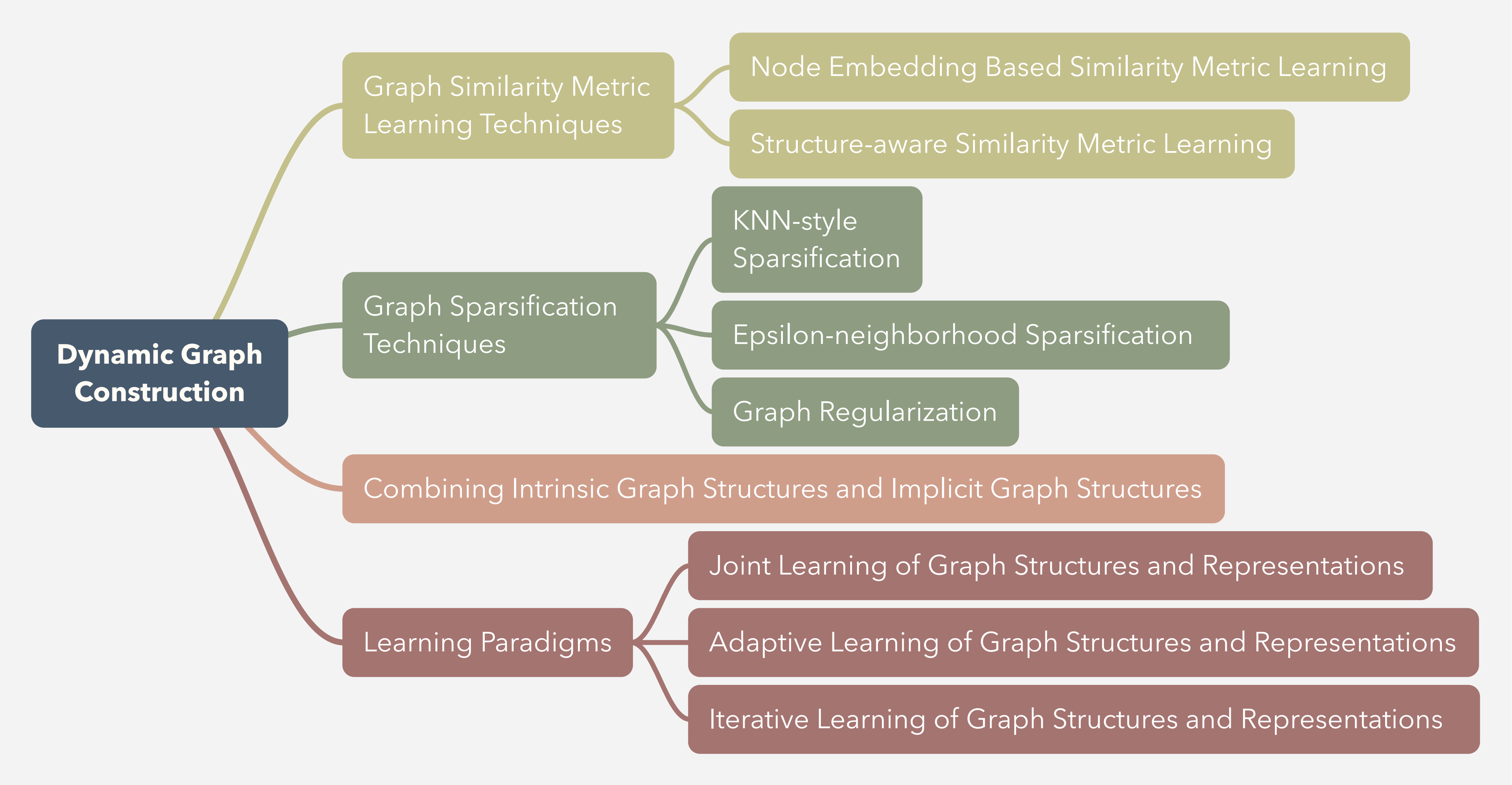
Learning Paradigms: Iterative Learning



Repeat until condition satisfied

Chen et al. “Iterative Deep Graph Learning for Graph Neural Networks: Better and Robust Node Embeddings”. NeurIPS 2021.

Dynamic Graph Construction Summary



Graph4NLP

Static vs. Dynamic Graph Construction

New topic in DLG4NLP!

Static graph construction	Dynamic graph construction
Pros	Pros
prior knowledge	no domain expertise
	joint graph structure & representation learning
Cons	Cons
extensive domain expertise	scalability
• error-prone (e.g., noisy, incomplete) • sub-optimal	explainability
• disjoint graph structure & representation learning • error accumulation	

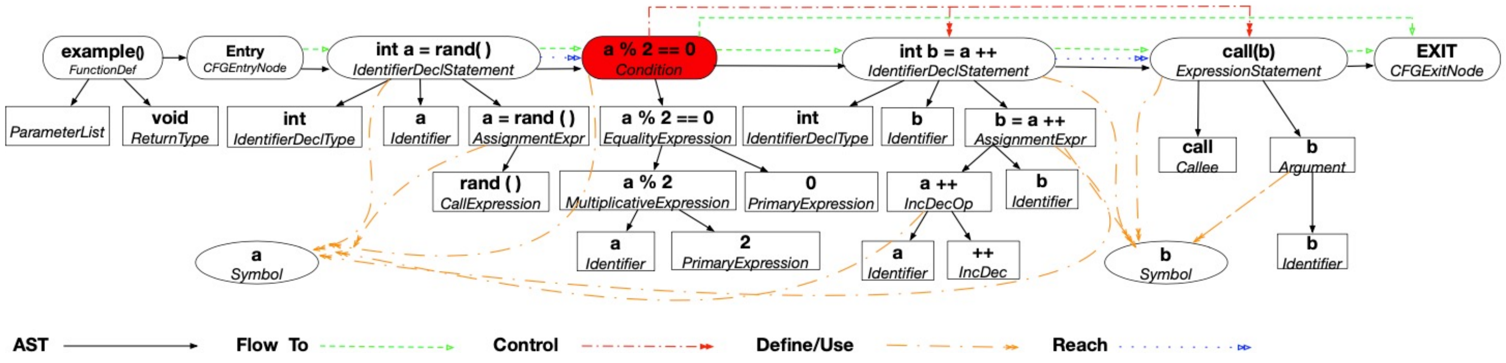
Static vs. Dynamic Graph Construction (cont)

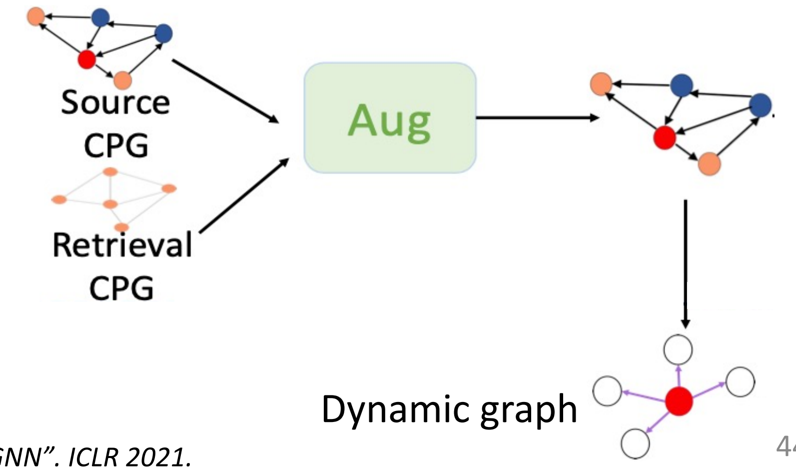
When to use static graph construction

- Domain knowledge which fits the task and can be presented as a graph

When to use dynamic graph construction

- Lack of domain knowledge which fits the task or can be presented as a graph
- Domain knowledge is incomplete or might contain noise
- To learn implicit graph which augments the static graph

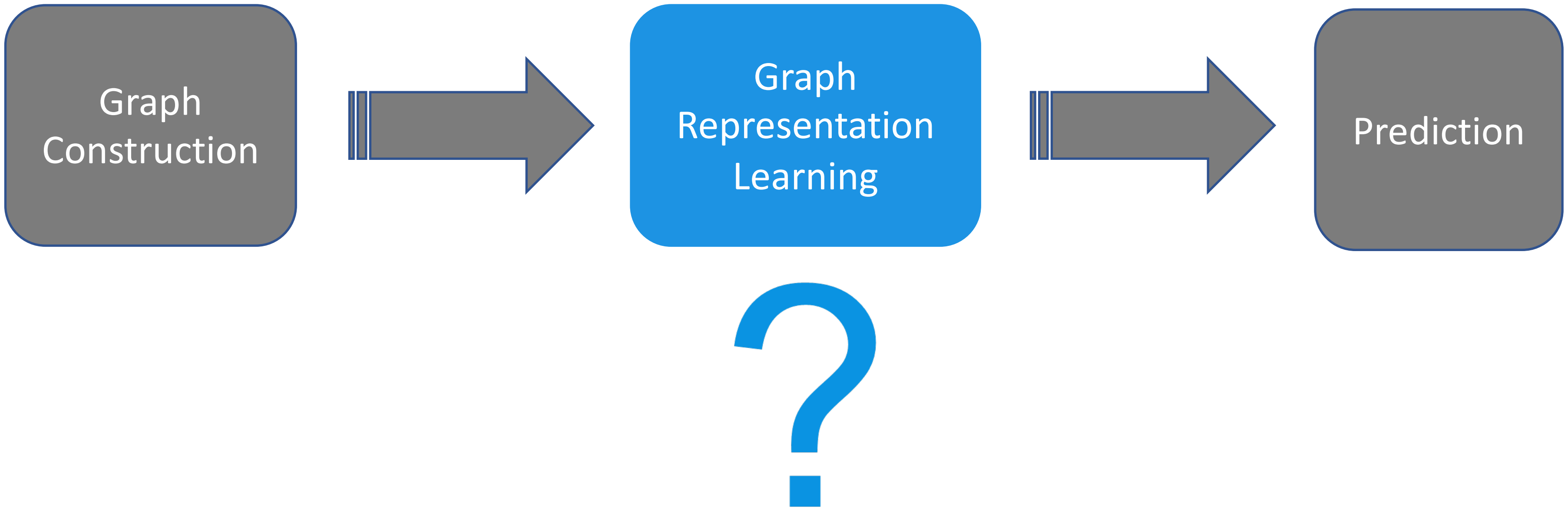




Liu et al. "Retrieval-Augmented Generation for Code Summarization via Hybrid GNN". ICLR 2021.

Graph Representation Learning for NLP

GNNs for Graph Representation Learning

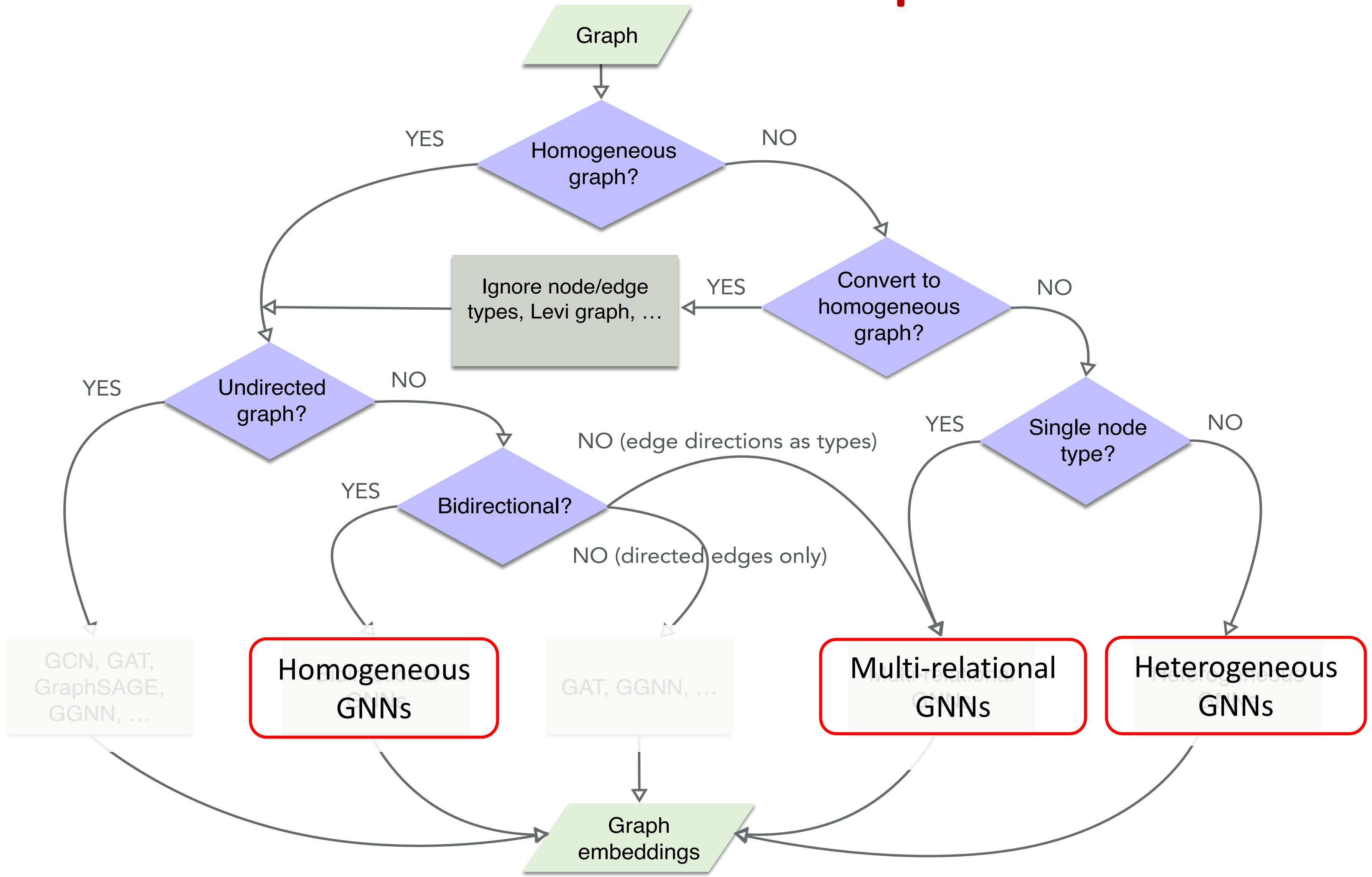


Homogeneous vs Multi-relational vs Heterogeneous Graphs

Graph types	Homogeneous	Multi-relational	Heterogeneous
# of node types	1	1	> 1
# of edge types	1	> 1	>= 1

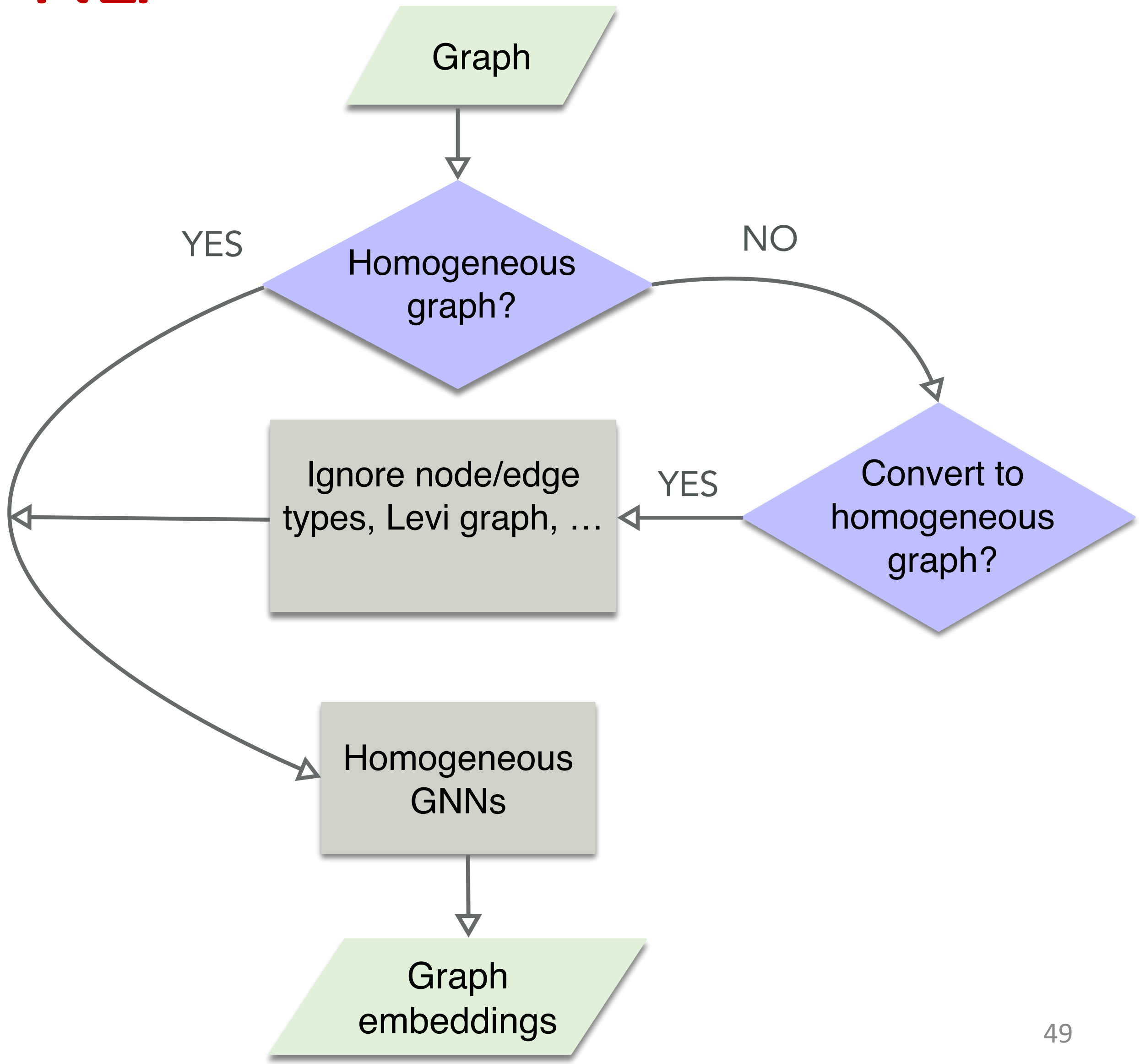
Graph4NLP

Which GNNs to Use Given a Graph?

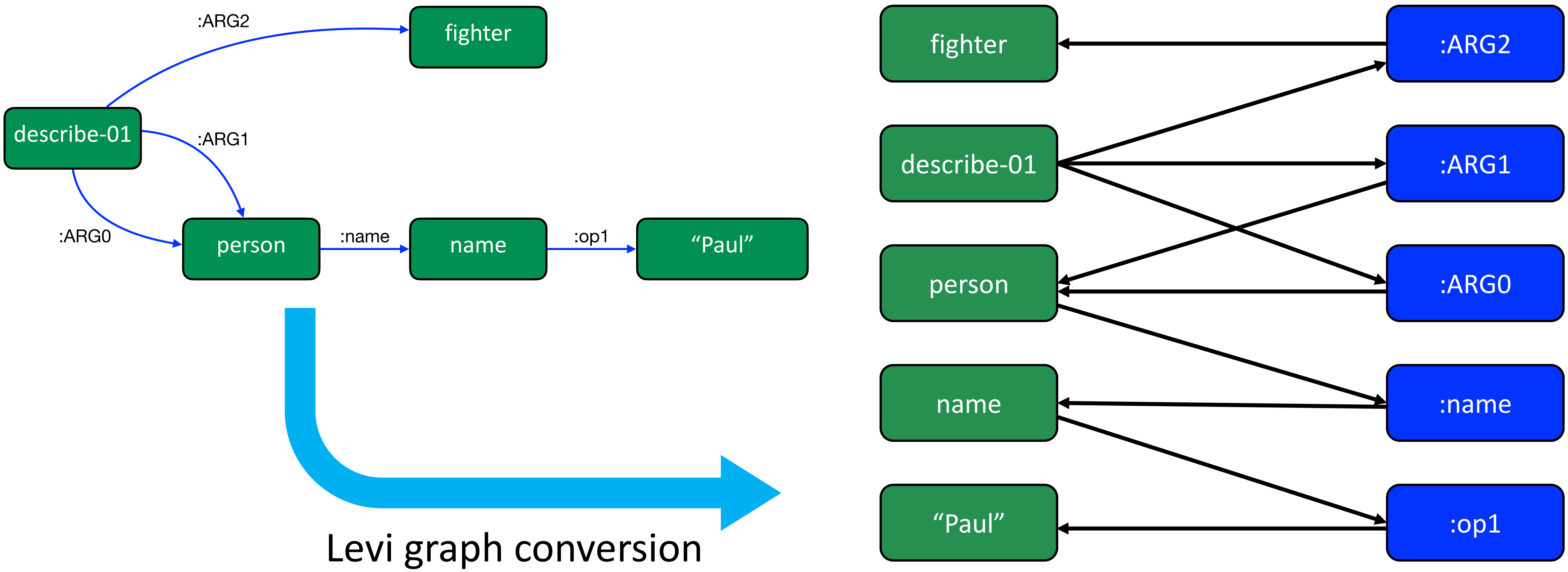


Homogeneous GNNs for NLP

- When to use homogeneous GNNs?
- Homogeneous GNNs
  - GCN
  - GAT
  - GraphSAGE
  - GGNN
  - …



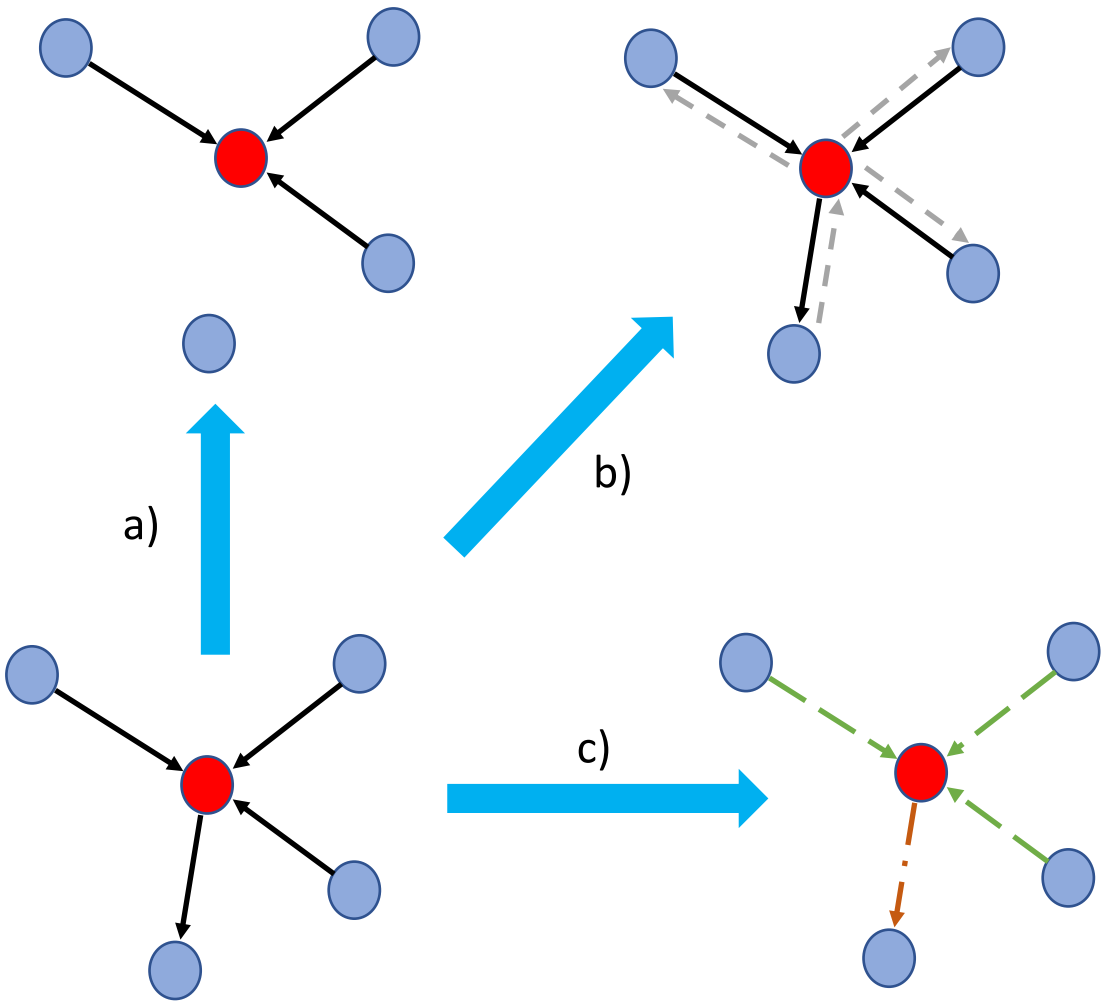
Non-homogeneous to Homogeneous Conversion via Levi Graph



Levi graph: edges as new nodes

How to Handle Edge Direction Information?

- Edge direction is important (think about BiLSTM, BERT)
- Common strategies for handling directed graphs
  - a) Message passing only along directed edges (e.g., GAT, GGNN)
  - b) Regarding edge directions as edge types (i.e., adding “reverse” edges)
  - c) Bidirectional GNNs



Edge Directions as Edge Types

- Regarding edge directions as edge types, resulting in a multi-relational graph

$$dir_{i,j} = \begin{cases} default, & e_{i,j} \text{ is originally existing in the graph} \\ inverse, & e_{i,j} \text{ is the inverse edge} \\ self, & i = j \end{cases}$$

Then we can apply multi-relational GNNs

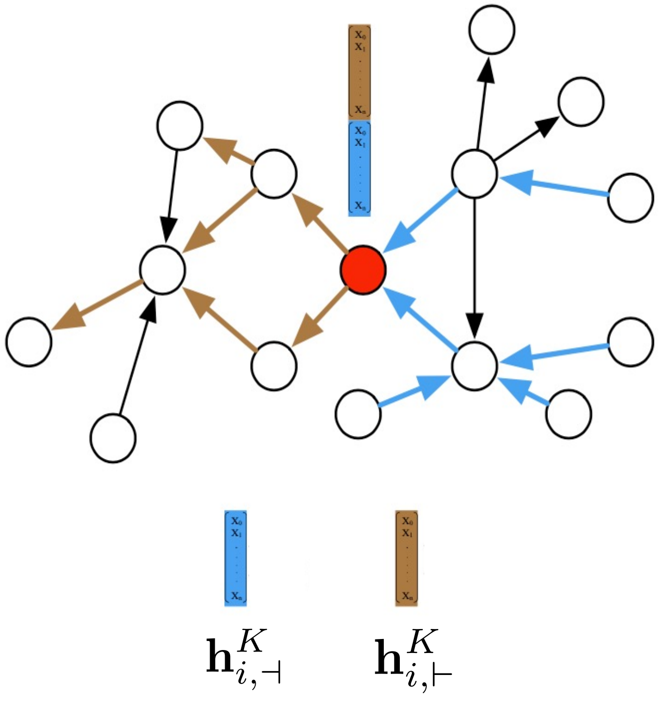
Bidirectional GNNs for Directed Graphs

Bi-Sep GNNs formulation:

Run multi-hop backward/forward GNN on the graph

$$\mathbf{h}_{i,\dashv}^{k} = GNN(\mathbf{h}_{i,\dashv}^{k-1}, \{\mathbf{h}_{j,\dashv}^{k-1} : \forall v_j \in \mathcal{N}_{\dashv}(v_i)\})$$

$$\mathbf{h}_{i,\vdash}^{k} = GNN(\mathbf{h}_{i,\vdash}^{k-1}, \{\mathbf{h}_{j,\vdash}^{k-1} : \forall v_j \in \mathcal{N}_{\vdash}(v_i)\})$$



Concatenate backward/forward node embeddings at last hop

$$\mathbf{h}_{i}^{K} = \mathbf{h}_{i,\dashv}^{K} || \mathbf{h}_{i,\vdash}^{K}$$

Xu et al. "Graph2Seq: Graph to Sequence Learning with Attention-based Neural Networks". 2018.

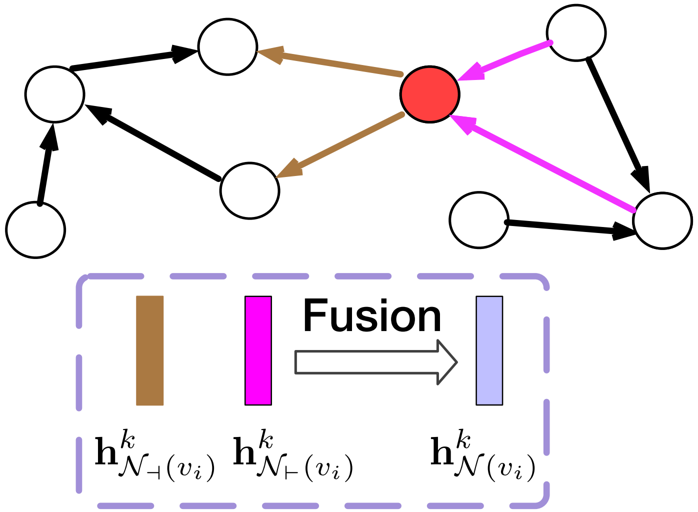
Bidirectional GNNs for Directed Graphs (cont)

Bi-Fuse GNNs formulation:

Run one-hop backward/forward node aggregation

$$\mathbf{h}^k_{\mathcal{N}_{\dashv}(v_i)} = AGG(\mathbf{h}^{k-1}_i, \{\mathbf{h}^{k-1}_j : \forall v_j \in \mathcal{N}_{\dashv}(v_i)\})$$

$$\mathbf{h}^k_{\mathcal{N}_{\vdash}(v_i)} = AGG(\mathbf{h}^{k-1}_i, \{\mathbf{h}^{k-1}_j : \forall v_j \in \mathcal{N}_{\vdash}(v_i)\})$$



Fuse backward/forward aggregation vectors at each hop

$$\mathbf{h}^k_{\mathcal{N}(v_i)} = Fuse(\mathbf{h}^k_{\mathcal{N}_{\dashv}(v_i)}, \mathbf{h}^k_{\mathcal{N}_{\vdash}(v_i)})$$

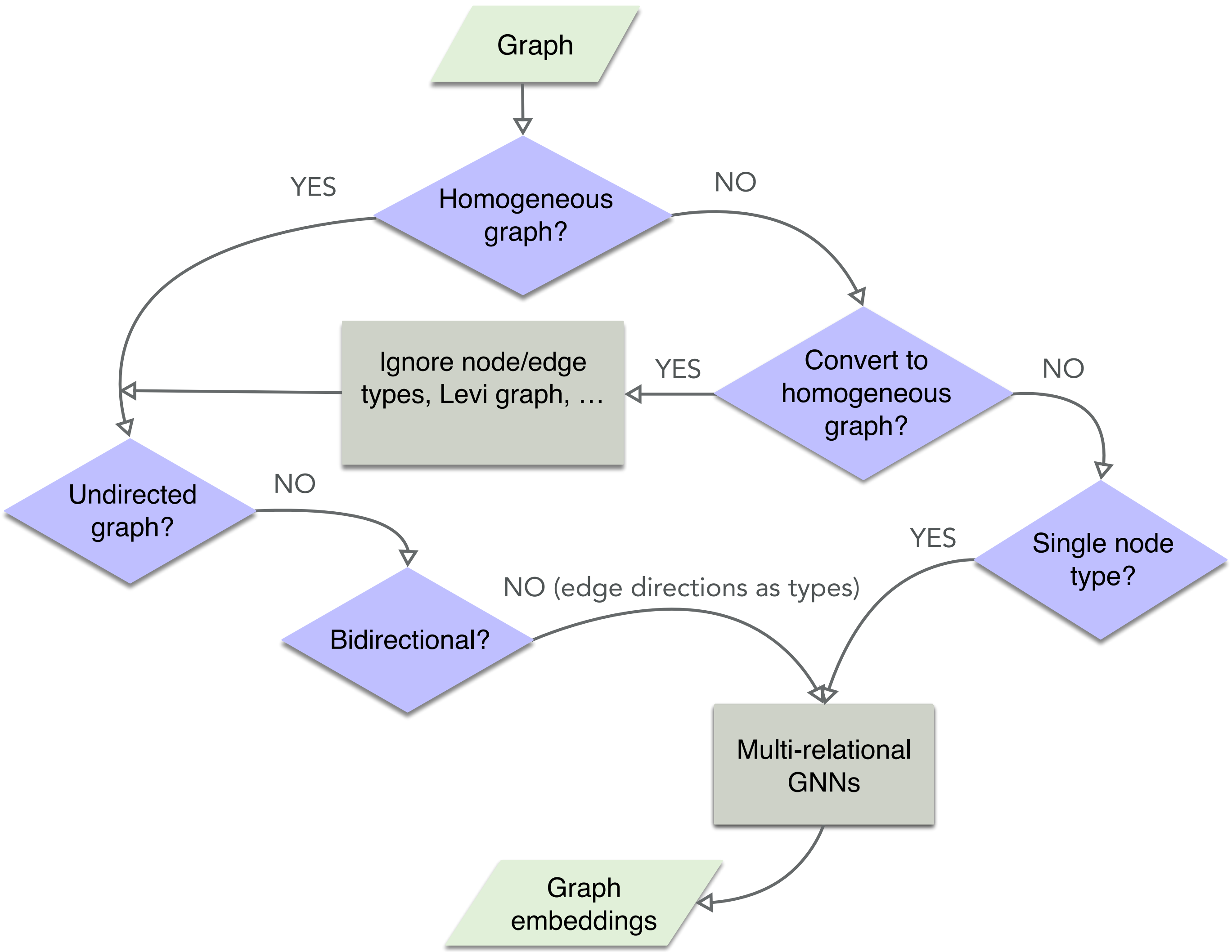
Update node embeddings with fused aggregation vectors at each hop

$$\mathbf{h}^k_i = \sigma(\mathbf{h}^{k-1}_i, \mathbf{h}^k_{\mathcal{N}(v_i)})$$

Chen et al. “Reinforcement Learning Based Graph-to-Sequence Model for Natural Question Generation”. ICLR 2020.

Multi-relational GNNs for NLP

- When to use multi-relational GNNs?
- Multi-relational GNNs
    a) Including relation-specific transformation parameters in GNN
    b) Including edge embeddings in GNN
    c) Multi-relational Graph Transformers



R-GNN: Overview

$$\mathbf{h}_i^k = \sigma(\mathbf{h}_i^{k-1}, \sum_{v_j \in \mathcal{N}(v_i)} AGG(\mathbf{h}_j^{k-1}, \theta^k))$$

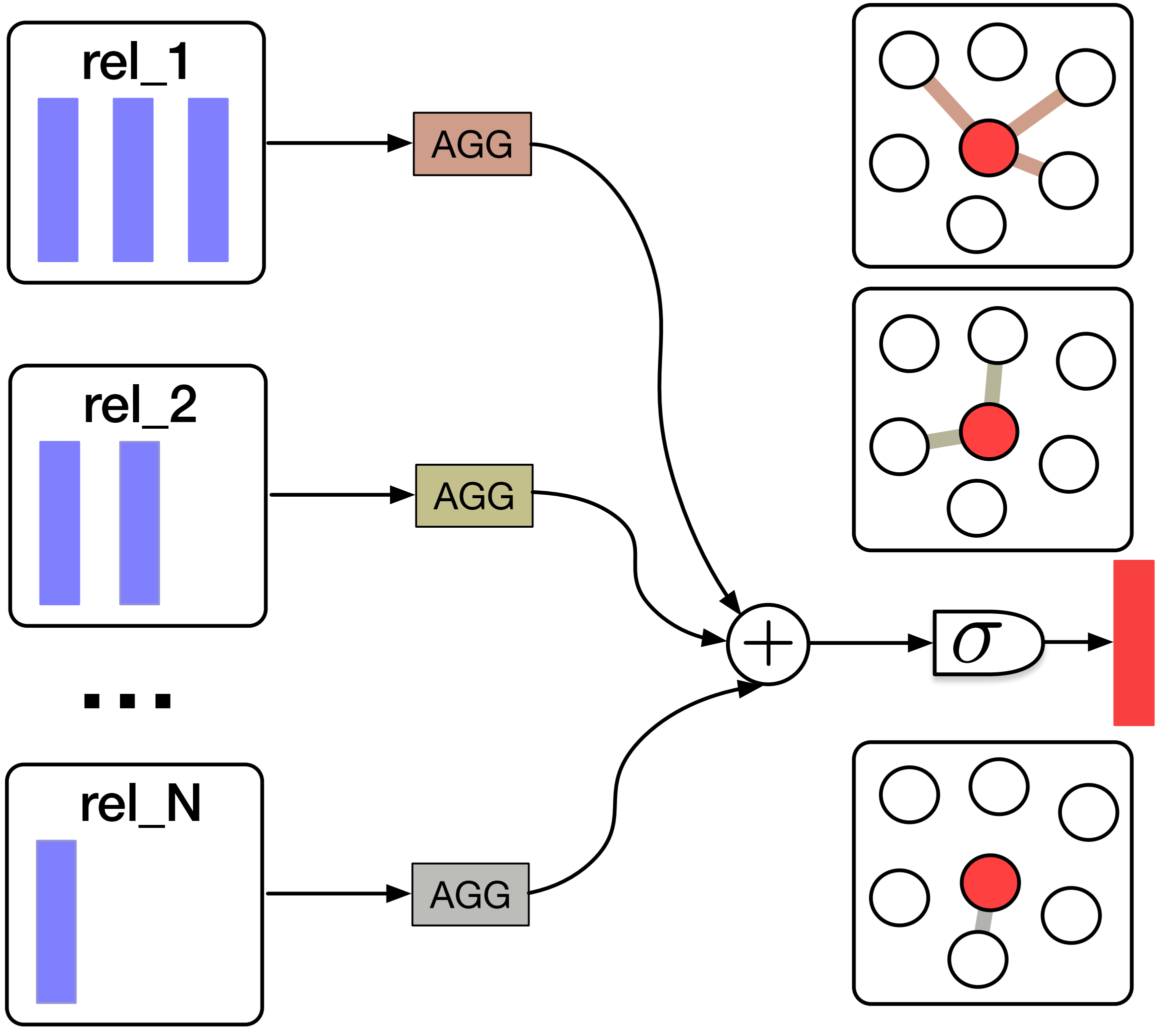
GNN

R-GNN

1) relation-specific transformation, e.g., node feature transformation, attention weight …

$$\mathbf{h}_i^k = \sigma(\mathbf{h}_i^{k-1}, \sum_{r \in \mathcal{E}} \sum_{v_j \in \mathcal{N}_r(v_i)} AGG(\mathbf{h}_j^{k-1}, \theta_r^k))$$

2) aggregation per relation-specific subgraph



R-GNN Variant: R-GCN

- Relation-specific node feature transformation during neighborhood aggregation

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

Relation-specific d x d learnable weight matrix

Schlichtkrull et al. "Modeling Relational Data with Graph Convolutional Networks". 2017.

R-GNN: Avoiding Over-parameterization

Learning d x d transformation weight matrix for each relation is expensive!

O(Rd^2) parameters every GNN layer where R is the num of relation types

How to avoid over-parameterization?

Option 1) basis decomposition - linear hypothesis

$$\theta_r^k = \sum_{b=1}^{B} a_{rb}^k \mathbf{V}_b^k, \quad \mathbf{V}_b^{(k)} \in \mathbb{R}^{d \times d}$$

O(RB + Bd^2) parameters

Basis matrices

Option 2) block-diagonal decomposition - sparsity hypothesis

$$\theta_r^k = \bigoplus_{b=1}^{B} \mathbf{Q}_{br}^k = diag(\mathbf{Q}_{1r}^k, \mathbf{Q}_{2r}^k, ..., \mathbf{Q}_{Br}^k), \quad \mathbf{Q}_{br}^{(k)} \in \mathbb{R}^{d/B \times d/B}$$

O(Rd^2/B) parameters

Submatrices

Including Edge Embeddings in GNNs

Variant 1) Include edge embeddings in message passing

$$\mathbf{h}_i^k = \sigma(\mathbf{h}_i^{k-1}, \sum_{v_j \in \mathcal{N}(v_i)} AGG(\mathbf{h}_j^{k-1}, \mathbf{e}_{i,j}, \theta^k))$$

Edge embeddings

Variant 2) Update edge embedding in message passing

$$\mathbf{h}_i^k = \sigma(\mathbf{h}_i^{k-1}, \sum_{v_j \in \mathcal{N}(v_i)} AGG(\mathbf{h}_j^{k-1}, \mathbf{e}_{i,j}^{k-1}, \theta^k)), \quad \mathbf{e}_{i,j}^k = f(\mathbf{e}_{i,j}^{k-1}, \theta_{rel}^k)$$

Update edge embeddings

Chen et al. "Toward Subgraph Guided Knowledge Graph Question Generation with Graph Neural Networks". 2020.
Vashishth et al. "Composition-based Multi-Relational Graph Convolutional Networks". ICLR 2020.

Multi-relational Graph Transformers

- Transformers as a special class of GNNs which
  - jointly learn and encode a fully-connected graph via self-attention
  - share many similarities with GAT
  - fail to effectively handle arbitrary graph-structured data
    - e.g., position embeddings for sequential data, removing position embeddings for set
- Multi-relational graph transformers
  - employed with structure-aware self-attention
  - respect various relation types

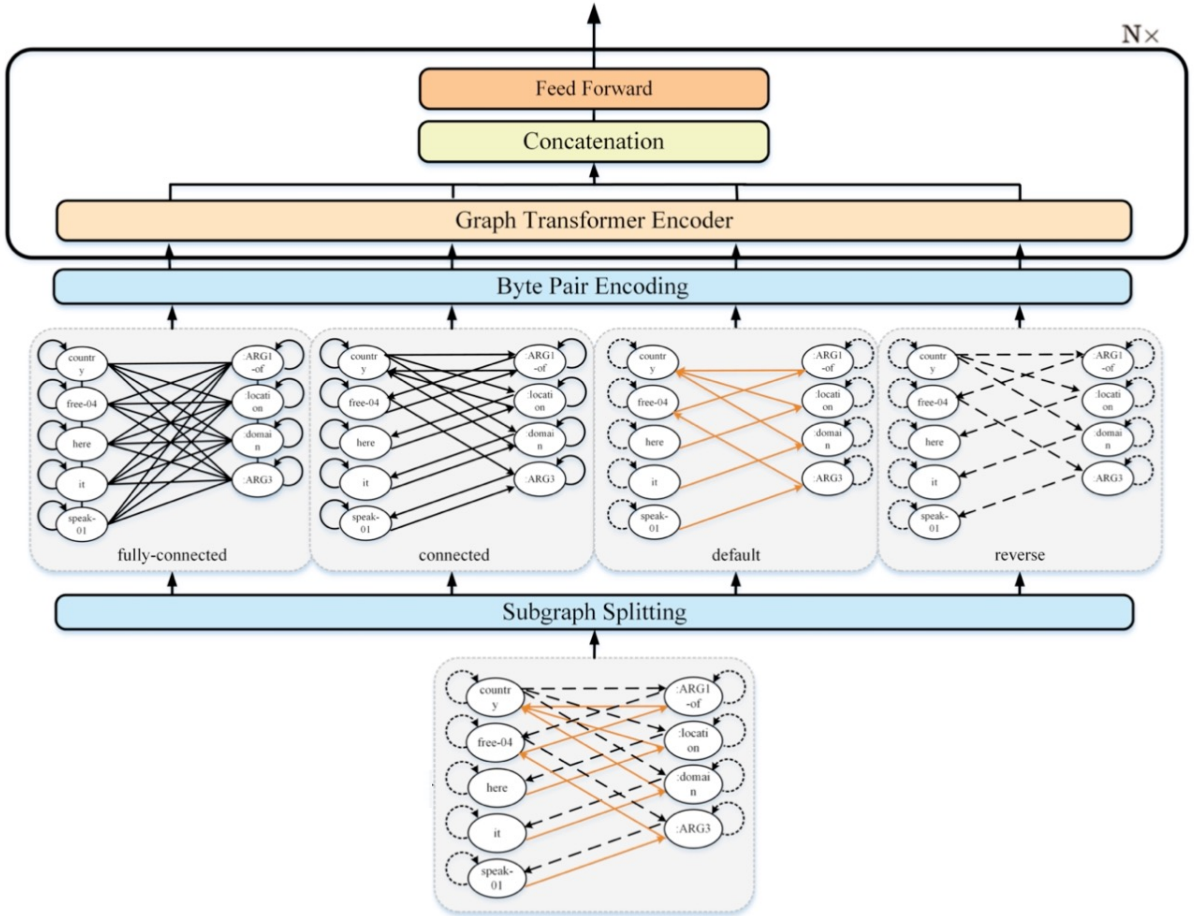
R-GAT based Graph Transformers

GAT-like masked attention

$$\mathbf{z}_i^{r,k} = \sum_{v_j \in \mathcal{N}_r(v_i)} \alpha_{i,j}^{k} \mathbf{W}_V^k \mathbf{h}_j^{k-1}, r \in \mathcal{E}$$

$$\mathbf{h}_i^k = \mathrm{FFN}^k(\mathbf{W}_O^k[\mathbf{z}_i^{R_1,k}, ..., \mathbf{z}_i^{R_m,k}])$$

Relation-specific learnable weight matrix



Yao et al. *"Heterogeneous Graph Transformer for Graph-to-Sequence Learning". ACL 2020.*

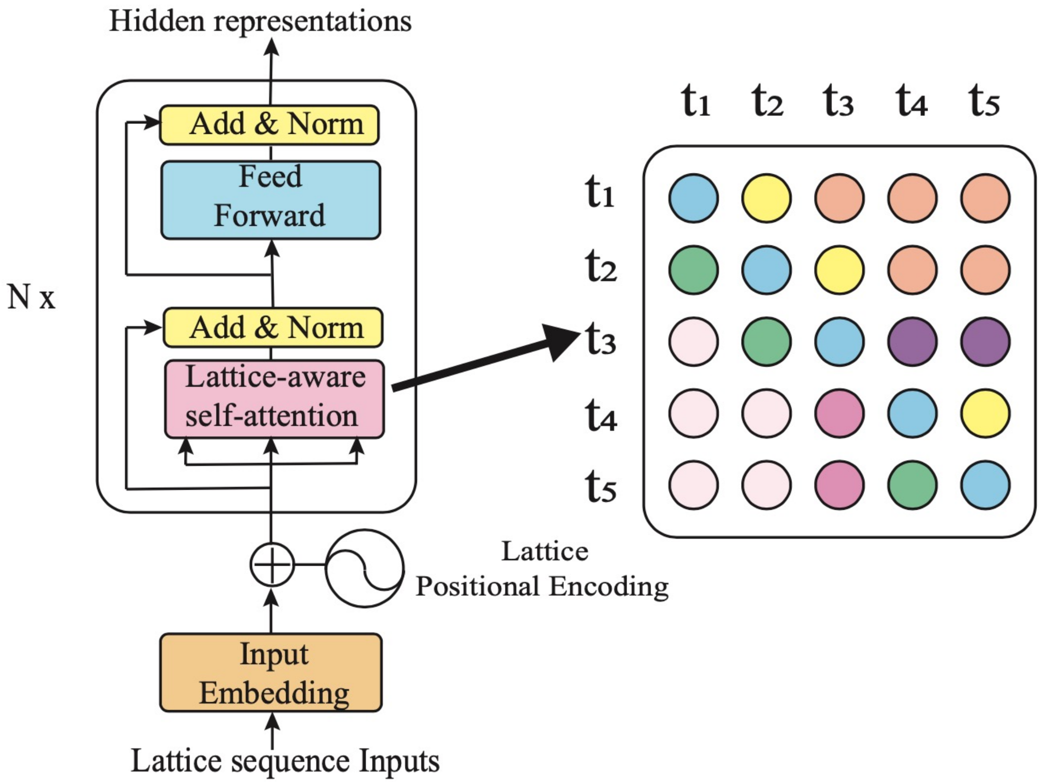
Structure-aware Self-attention based Graph Transformers

$$\mathbf{h}_i^k = \sum_j \alpha_{i,j}^k (\mathbf{W}_V^k \mathbf{h}_j^{k-1} + \mathbf{W}_F^k \mathbf{e}_{i,j})$$

$$\alpha_{i,j}^k = softmax(u_{i,j}^k)$$

$$u_{i,j}^k = \frac{(\mathbf{W}_Q^k \mathbf{h}_i^{k-1})^T (\mathbf{W}_K^k \mathbf{h}_j^{k-1} + \mathbf{W}_R^k \mathbf{e}_{i,j})}{\sqrt{d}}$$

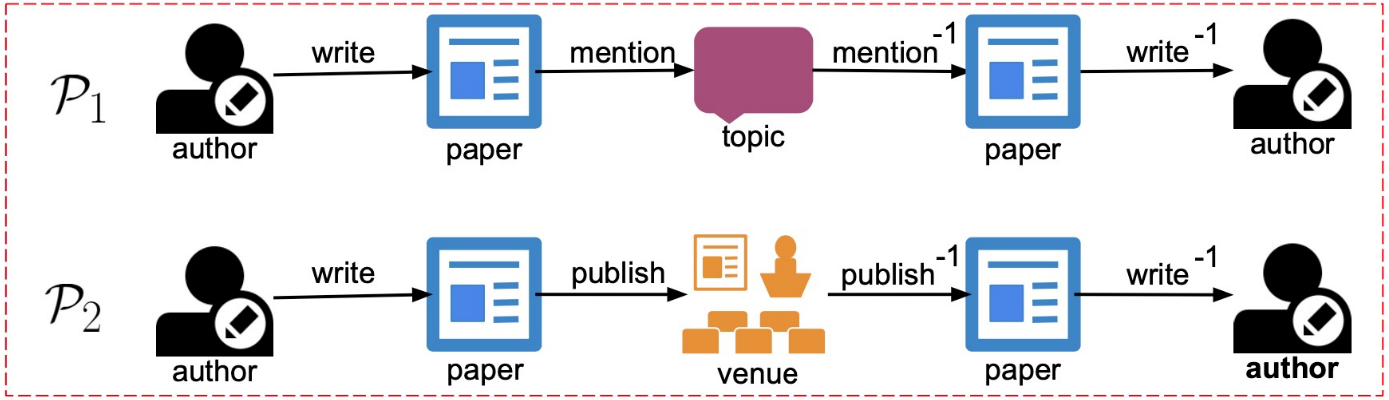
Edge embeddings



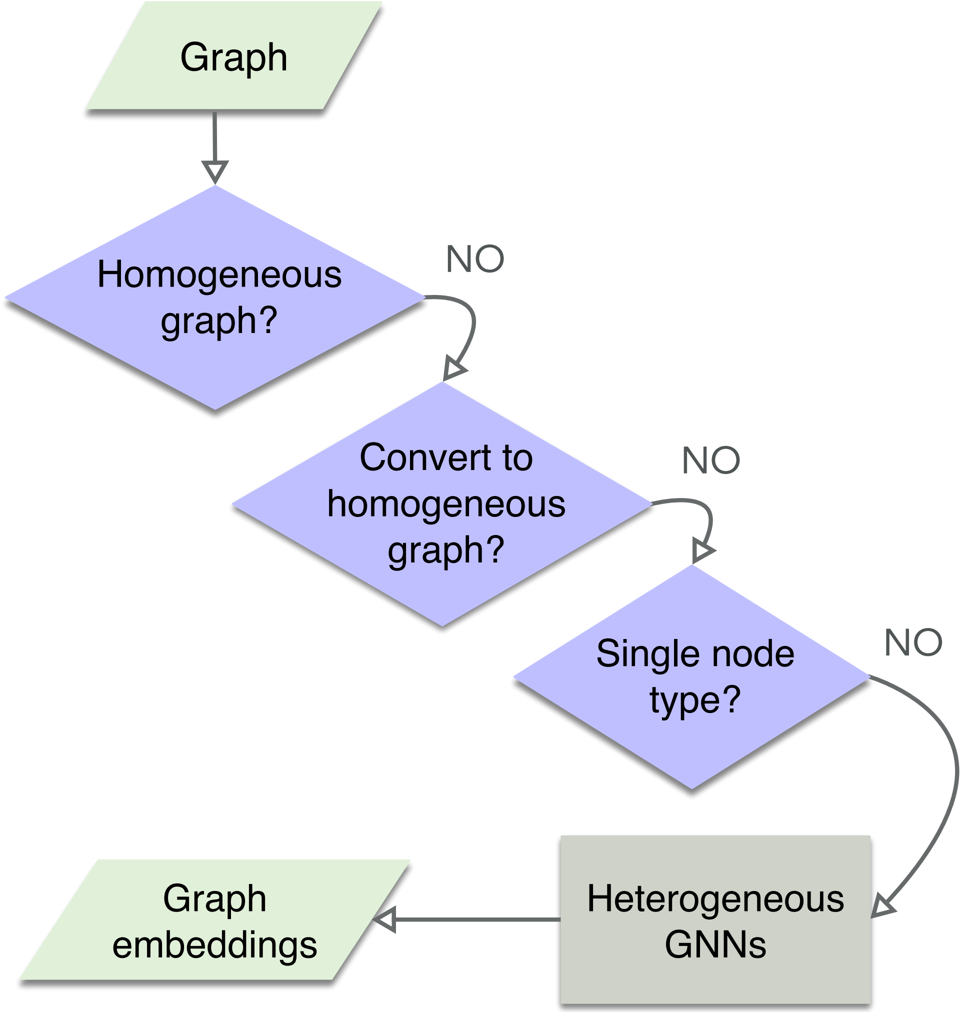
Xiao et al. "Lattice-Based Transformer Encoder for Neural Machine Translation". ACL 2019.
Zhu et al. "Modeling Graph Structure in Transformer for Better AMR-to-Text Generation". EMNLP 2019.

Heterogeneous GNNs

- When to use Heterogeneous GNNs?
- Heterogeneous GNNs
  a) Meta-path based Heterogeneous GNNs



Meta paths among author nodes



Meta-path based Heterogeneous GNN example: HAN

Step 1) type-specific node feature transformation

$$\mathbf{h}_i = \mathbf{W}_{\tau(v_i)} \mathbf{v}_i$$

Node-type specific learnable weight matrix

Step 2) node-level aggregation along each meta path

$$\mathbf{z}_{i,\Phi_k} = \sigma\Big(\sum_{v_j \in \mathcal{N}_{\Phi_k}(v_i)} \alpha_{i,j}^{\Phi_k} \mathbf{h}_j \Big)$$

Aggregate over neighboring nodes in k-length meta path

Step 3) meta-path level aggregation

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

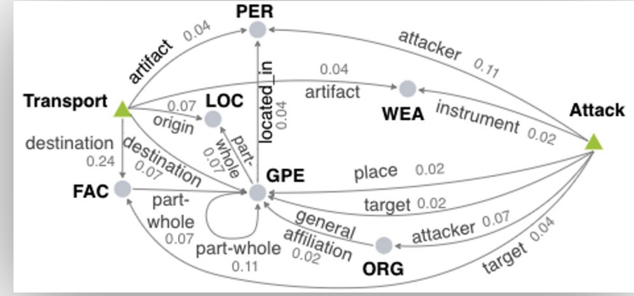
Attention weights over meta paths

Wang et al. "Heterogeneous Graph Neural Networks for Extractive Document Summarization". ACL 2020.

Graph Encoder-Decoder Models for NLP

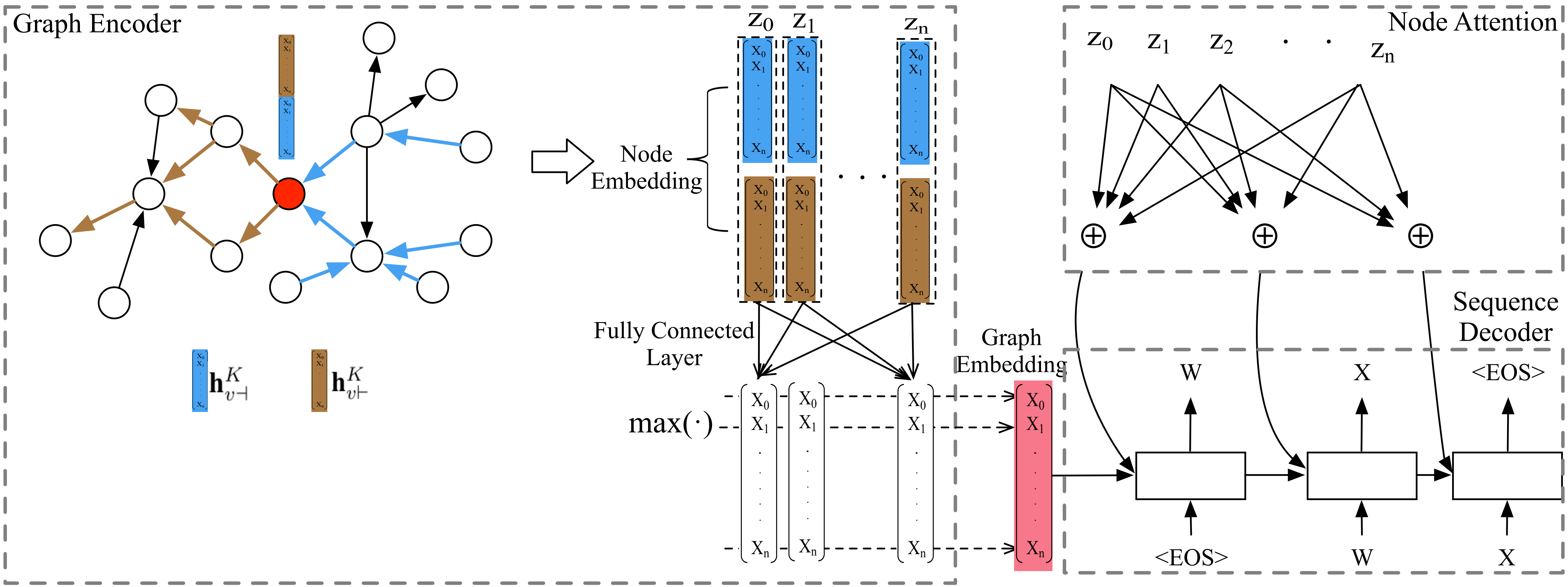
Seq2Seq: Applications and Challenges

- Applications
  - Machine translation
  - Natural language generation
  - Logic form translation
  - Information extraction



- Challenges
  - Only applied to problems whose inputs are represented as sequences
  - Cannot handle more complex structure such as graphs
  - Converting graph inputs into sequences inputs lose information
  - Augmenting original sequence inputs with additional structural information enhances word sequence feature

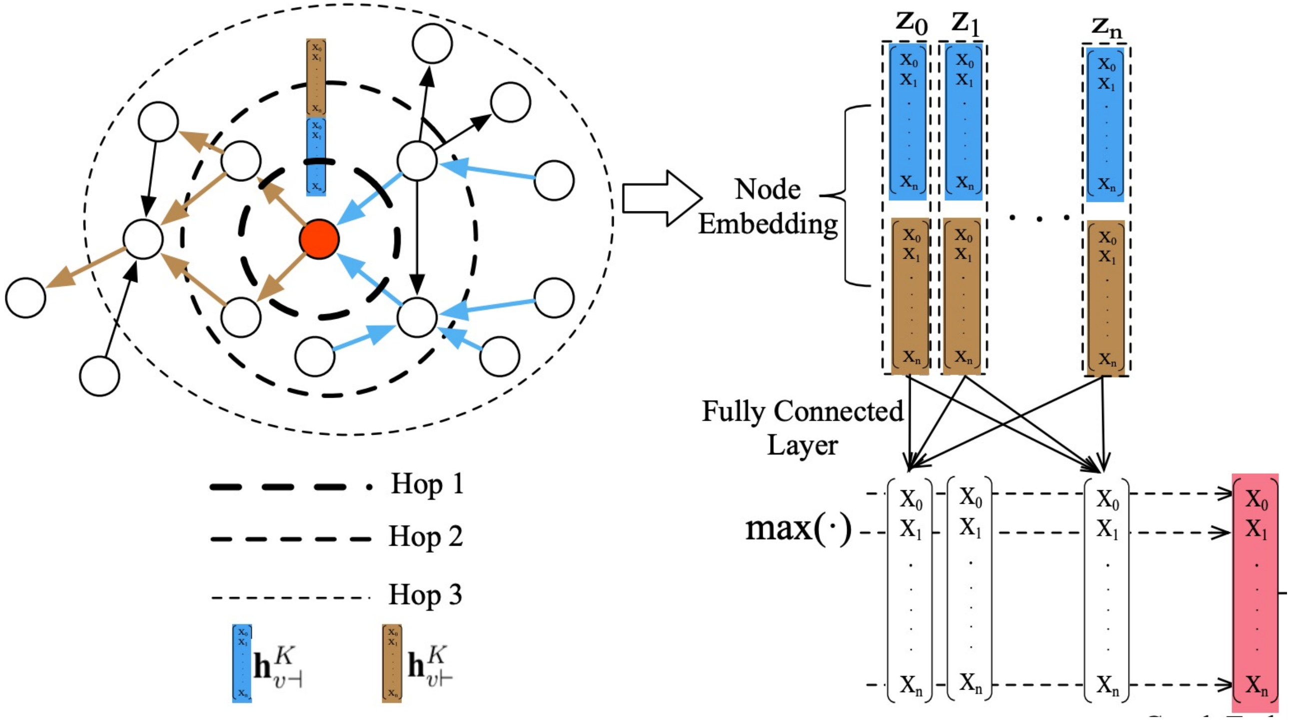
Graph-to-Sequence Model



[1] Kun Xu*, Lingfei Wu*, Zhiguo Wang, Yansong Feng, Michael Witbrock, and Vadim Sheinin (Equally Contributed), "Graph2Seq: Graph to Sequence Learning with Attention-based Neural Networks", arXiv 2018.
[2] Yu Chen, Lingfei Wu** and Mohammed J. Zaki (**Corresponding Author), "Reinforcement Learning Based Graph-to-Sequence Model for Natural Question Generation", ICLR'20.

Graph Encoding

- Graph embedding
  - Pooling based graph embedding (*max, min and average pooling*)
  - Node based graph embedding
    - ◻ Add one super node which is connected to all other nodes in the graph
    - ◻ The embedding of this super node is treated as graph embedding



Attention Based Sequence Decoding

$$c_i = \sum_{j=1}^{\mathcal{V}} \alpha_{ij} h_j, \ where \ \alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{\mathcal{V}} \exp(e_{ik})}, \ e_{ij} = a(s_{i-1}, h_j)$$

context vector

node representation

Attention Based Sequence Decoding

$$c_i = \sum_{j=1}^{\mathcal{V}} \alpha_{ij} h_j, \; where \; \alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{\mathcal{V}} \exp(e_{ik})}, \; e_{ij} = a(s_{i-1}, h_j)$$

context vector

node representation

attention weights

alignment model

Attention Based Sequence Decoding

$$c_i = \sum_{j=1}^{\mathcal{V}} \alpha_{ij} h_j, \ where\ \alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{\mathcal{V}} \exp(e_{ik})}, \ e_{ij} = a(s_{i-1}, h_j)$$

context vector

node representation

attention weights

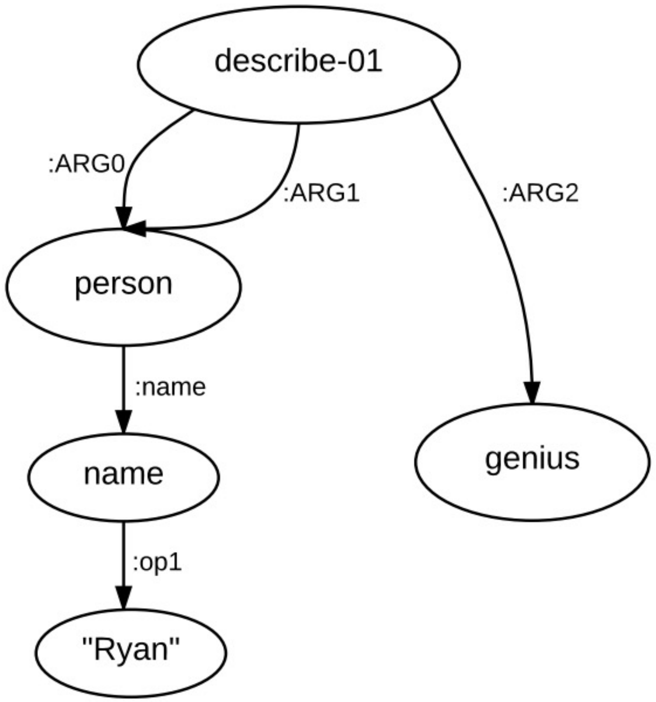
alignment model

- Objective Function

$$\theta^* = \arg\max_{\theta} \sum_{n=1}^{N} \sum_{t=1}^{T_n} \log p(y_t^n | y_{<t}^n, x^n)$$

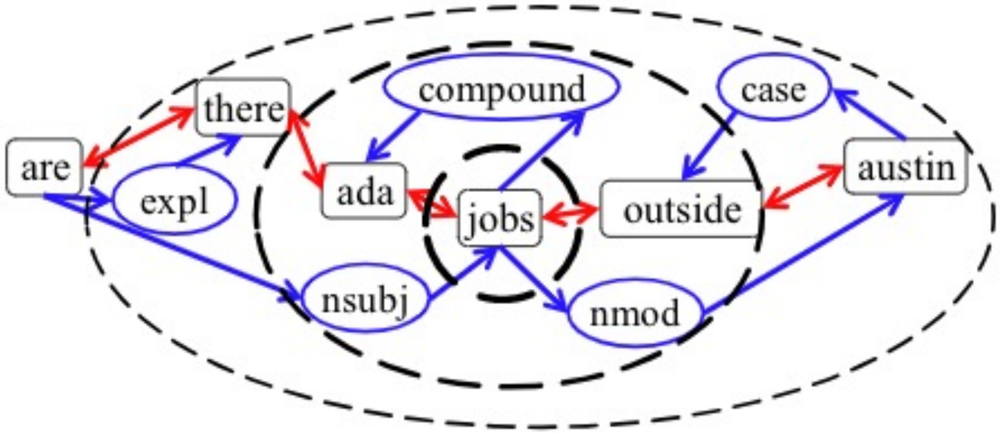
When Shall We Use Graph2Seq?

- Case I: the inputs are naturally or best represented in graph



"Ryan's description of himself: a genius."

- Case II: Hybrid Graph with sequence and its hidden structural information



Augmenting "are there ada jobs outside Austin" with its dependency parsing tree results

Learning Structured Input-Output Translation

- To bridge the semantic gap between the human-readable words and machine-understandable logics.
- Semantic parsing is important for question answering, text understanding
- Automatically solving of MWP is a growing interest.

SP	**Text Input:** what jobs are there for web developer who know 'c++' ?
	Structured output: answer(A , (job (A) , title (A , W) , const (W , 'Web Developer') , language (A , C) , const (C , 'c++')))
MWP	**Text input:** 0.5 of the cows are grazing grass . 0.25 of the cows are sleeping and 9 cows are drinking water from the pond . find the total number of cows .
	Structured output: ((0.5 * x) + (0.25 * x)) + 9.0 = x

Graph and Tree Constructions

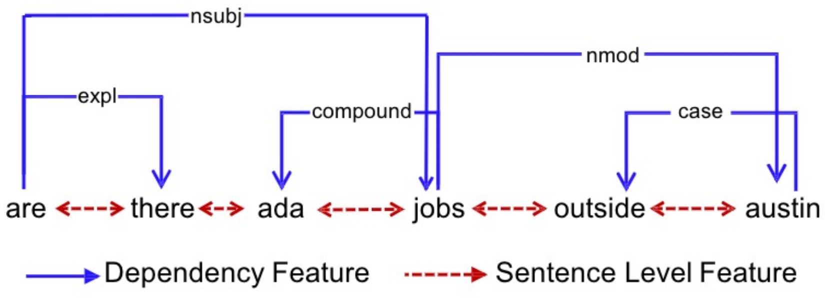


Figure 1: Dependency tree augmented text graph

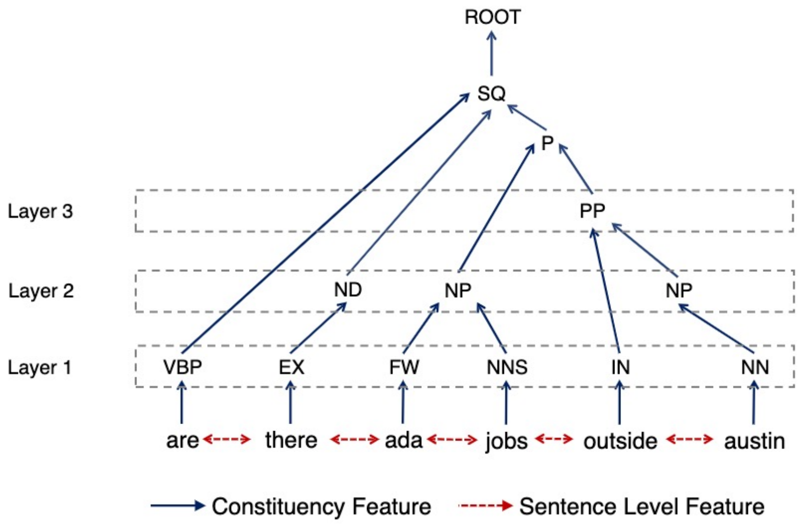


Figure 2: Constituency tree augmented text graph

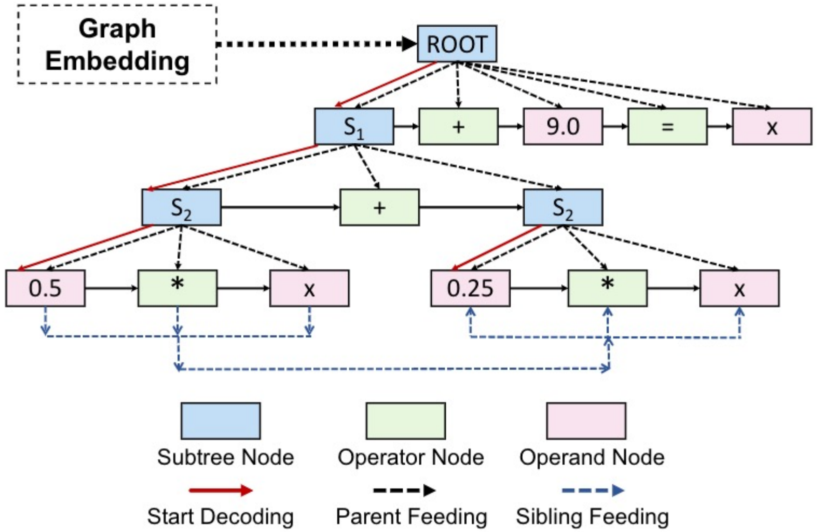
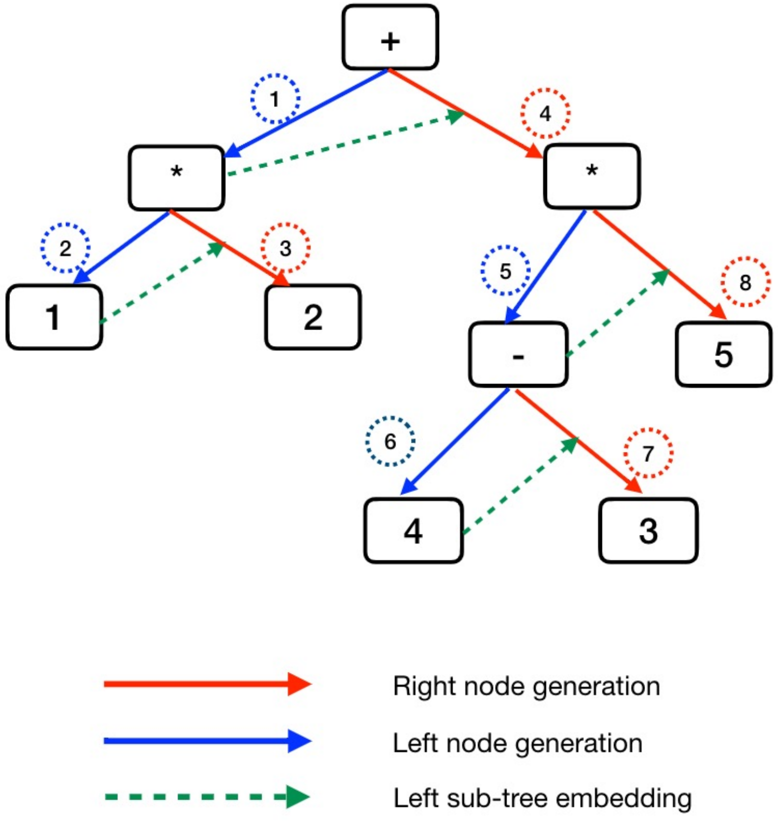
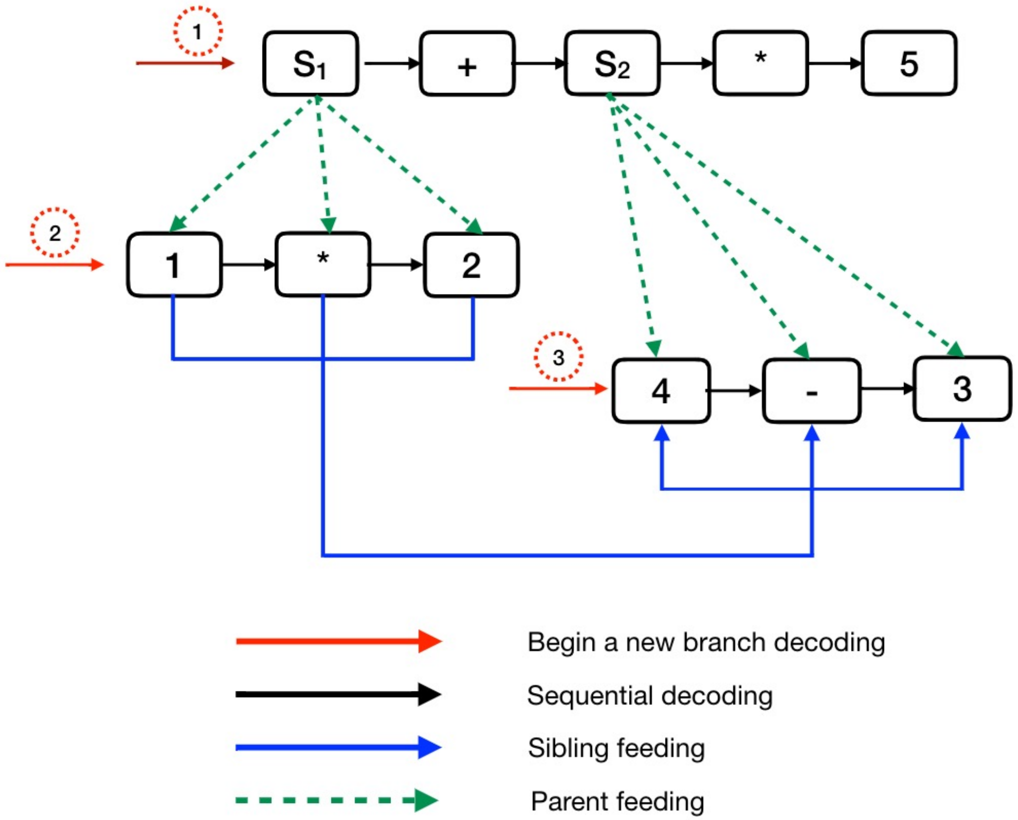


Figure 3: A sample tree output in our decoding process from expression "((**0.5** * **x**) + (**0.25** * **x**)) + **9.0** = **x**"

Tree Decoding

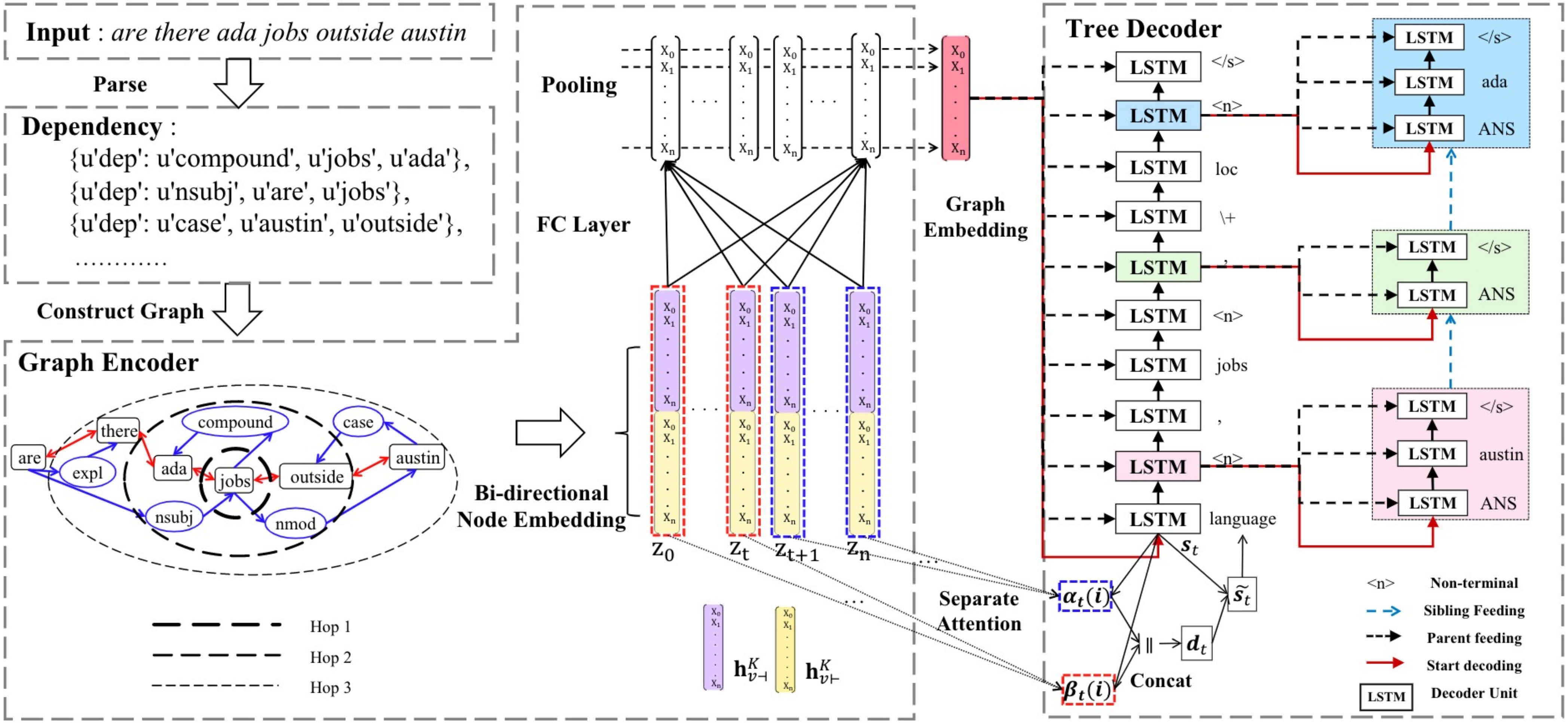


DFS-based tree decoder



BFS-based tree decoder

Graph-to-Tree Model



[1] Shucheng Li*, Lingfei Wu*, et al. "Graph-to-Tree Neural Networks for Learning Structured Input-Output Translation with Applications to Semantic Parsing and Math Word Problem", EMNLP 2020.

Separated Attention Based Tree Decoding

Context vector embeddings

$$\begin{cases} \mathbf{c}_{v_1} = \sum \alpha_{t(v)} \mathbf{z}_v, \forall v \in \mathcal{V}_1 \\ \mathbf{c}_{v_2} = \sum \beta_{t(v)} \mathbf{z}_v, \forall v \in \mathcal{V}_2 \end{cases}$$

Separated attention weights

$$\begin{cases} \alpha_{t(v)} = \dfrac{\exp(score(\mathbf{z}_v, \mathbf{s}_t))}{\exp(\sum_{k=1}^{V_1} score(\mathbf{z}_k, \mathbf{s}_t))}, \forall v \in \mathcal{V}_1 \\ \beta_{t(v)} = \dfrac{\exp(score(\mathbf{z}_v, \mathbf{s}_t))}{\exp(\sum_{k=1}^{V_2} score(\mathbf{z}_k, \mathbf{s}_t))}, \forall v \in \mathcal{V}_2 \end{cases}$$

Final attention hidden state

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

Outline

DLG4NLP Introduction

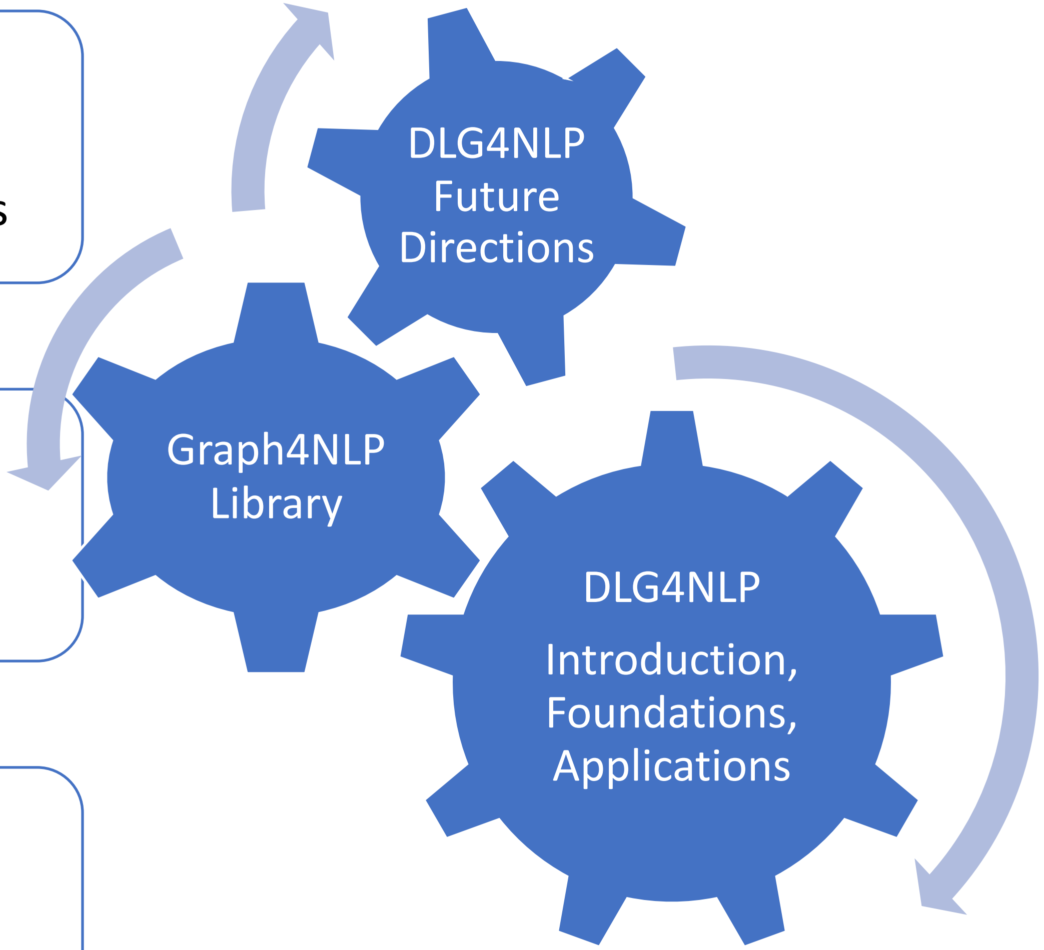
- Why Graphs for NLP?
- Conventional ML for NLP
- Deep Learning on Graphs: Foundations and Models

DLG4NLP Foundations

- Graph Construction for NLP
- Graph Representation Learning for NLP
- Graph Encoder-Decoder Models for NLP

DLG4NLP Applications

- Natural Question Generation
- Summarization



DLG4NLP Applications

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

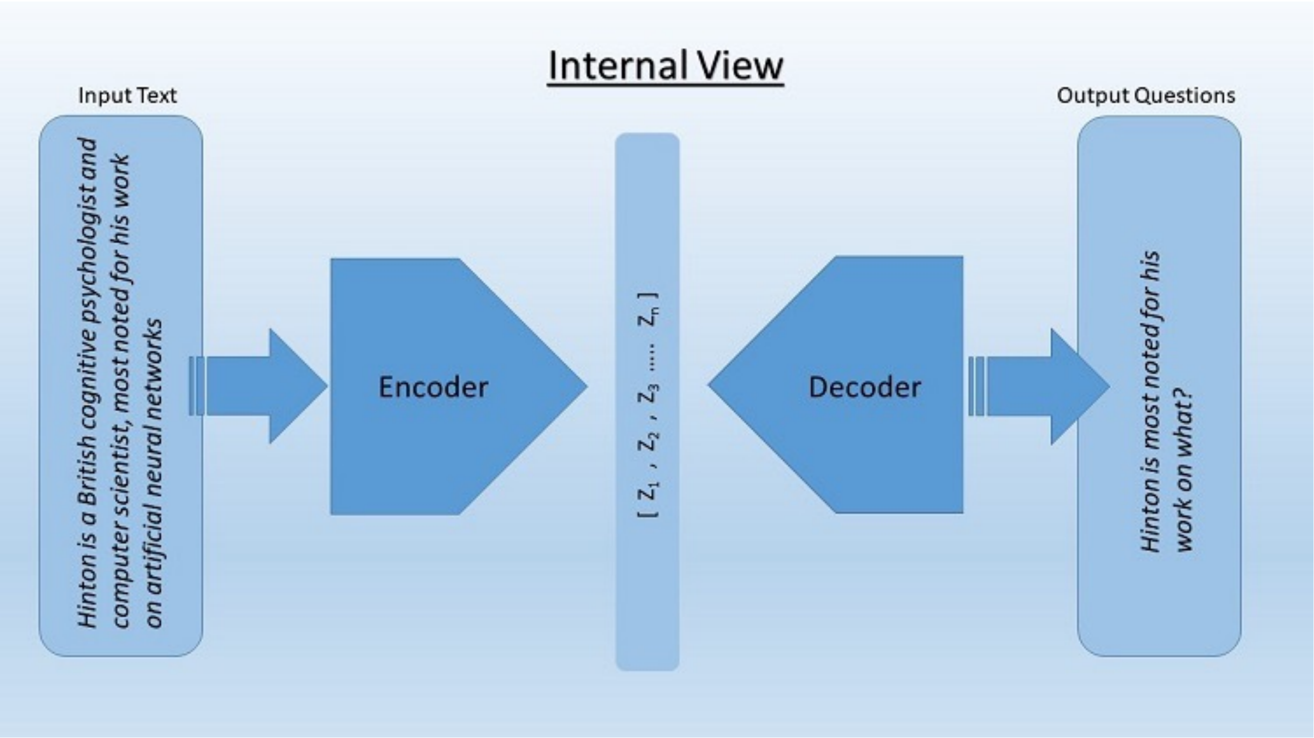
Graph4NLP

GNNs have been widely applied in various NLP tasks!

Wu, Chen et al, "Graph Neural Networks for Natural Language Processing: A Survey". *arxiv.org/abs/2106.06090*

Natural Question Generation

Natural Question Generation



- Input
  - A text passage $X^p = \{x_1^p, x_2^p, ..., x_N^p\}$
  - A target answer $X^a = \{x_1^a, x_2^a, ..., x_L^a\}$
- Output
  - A natural language question

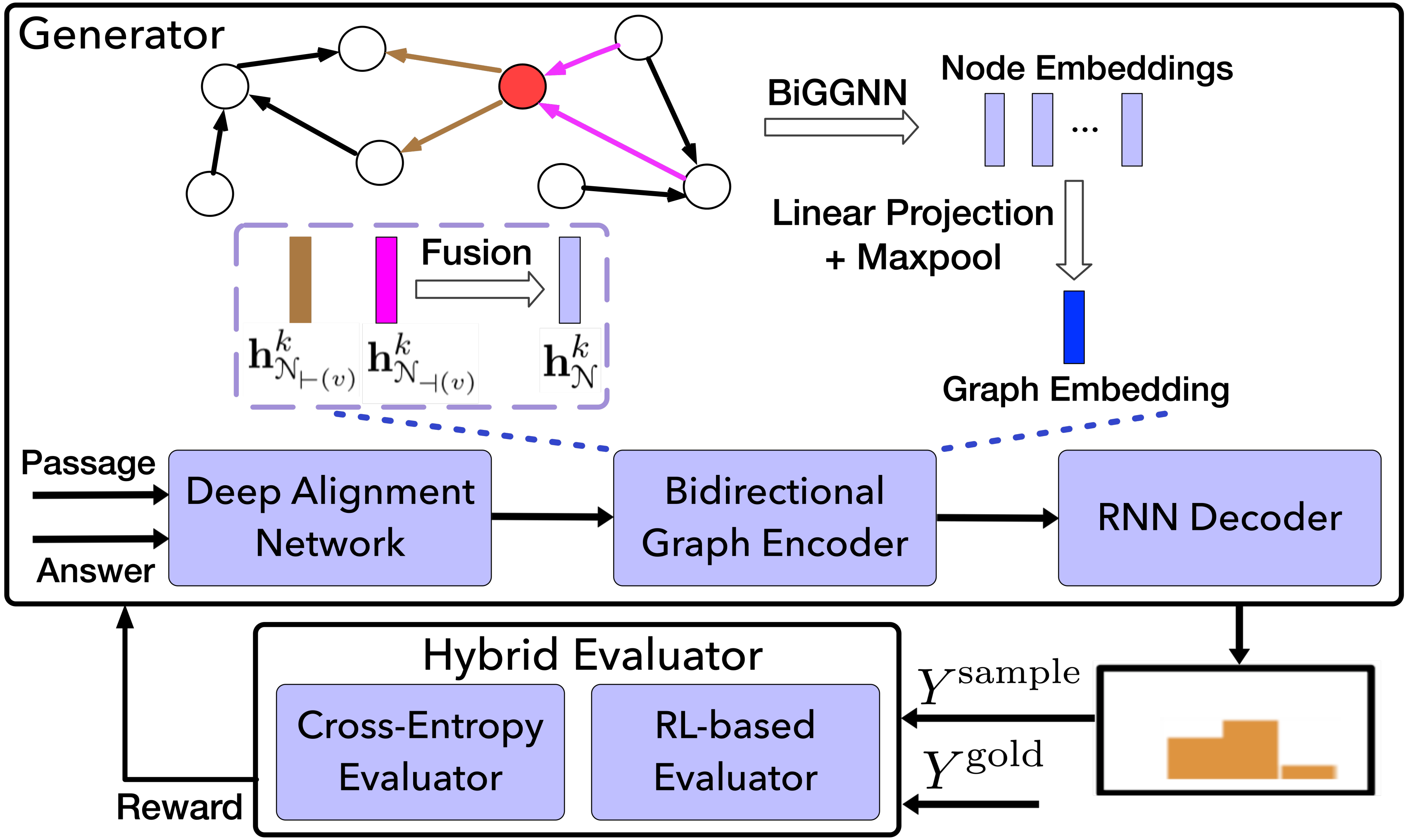
$$\hat{Y} = \{y_1, y_2, ..., y_T\}$$

which maximizes the conditional likelihood

$$\hat{Y} = \arg\max_Y P(Y|X^p, X^a)$$

Ref: https://images.app.goo.gl/TJEqwFS8nBNW8n8SA

RL-based Graph2Seq for QG [Chen et al. ICLR'20]

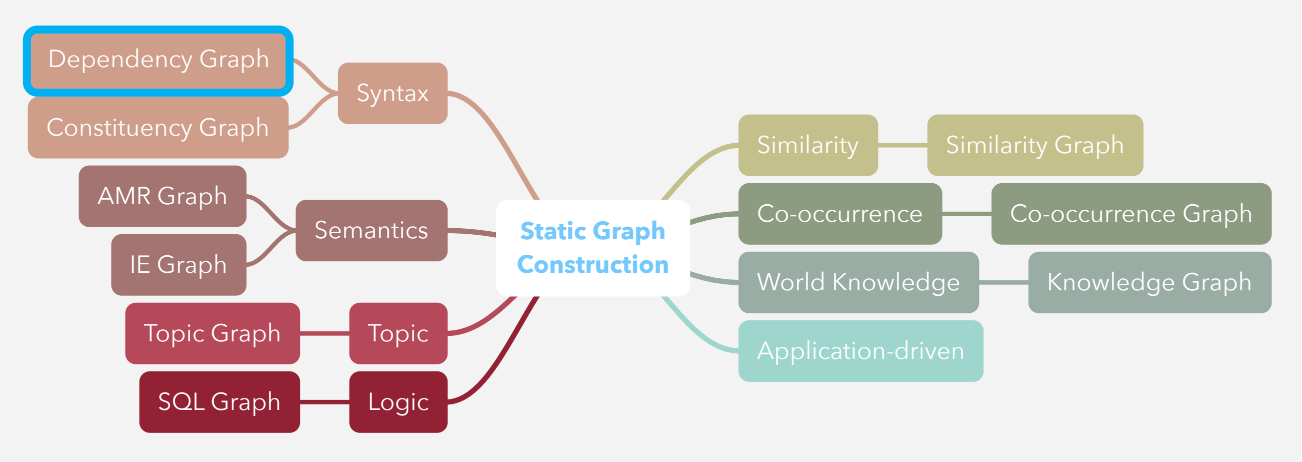


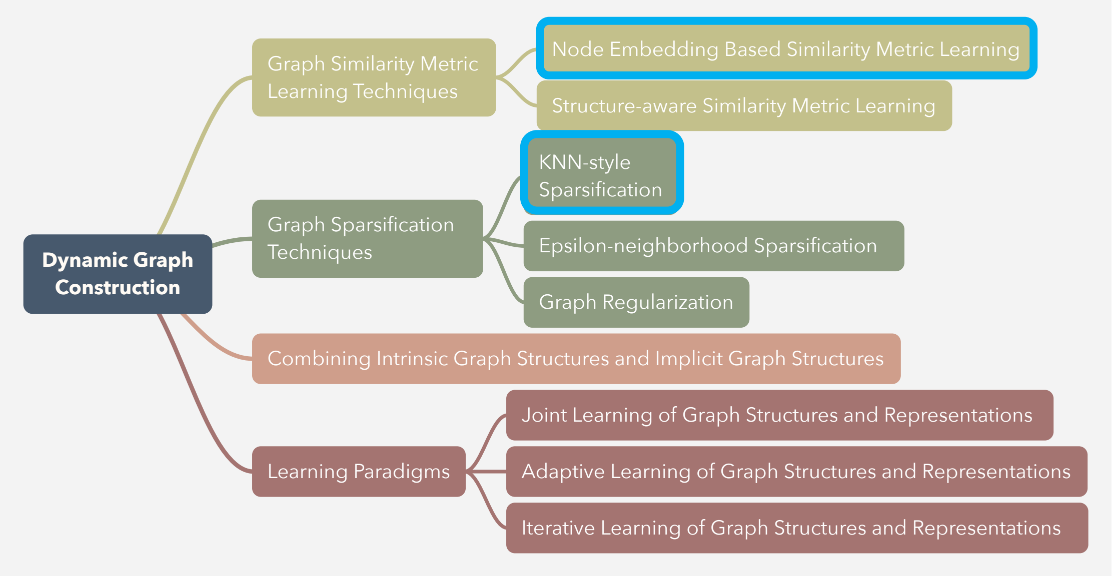
Chen et al. "Reinforcement Learning Based Graph-to-Sequence Model for Natural Question Generation". ICLR 2020.

RL-based Graph2Seq for QG [Chen et al. ICLR'20]

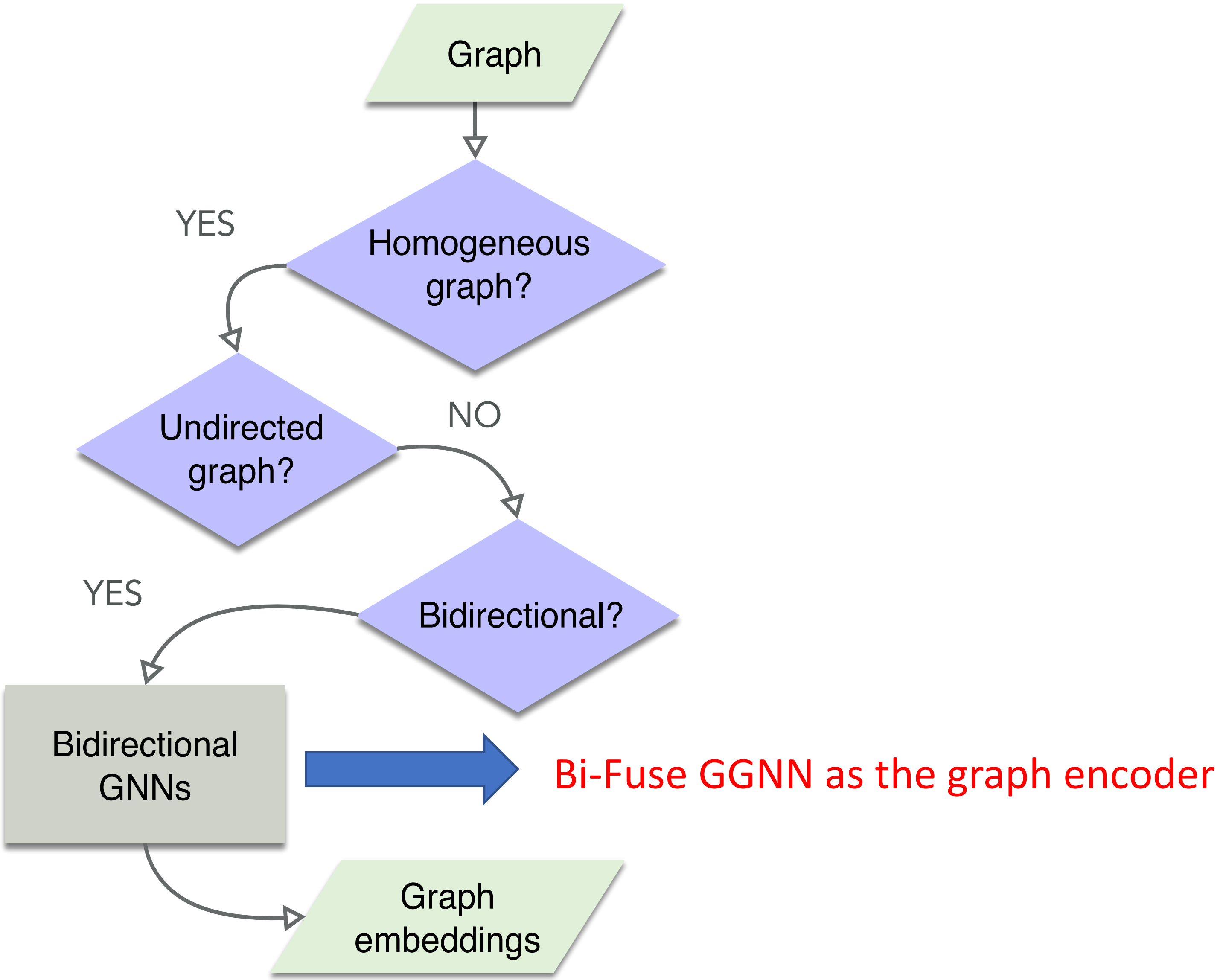
Two graph construction strategies:

1) Syntax-based static passage graph construction
2) Semantics-aware dynamic passage graph construction





RL-based Graph2Seq for QG [Chen et al. ICLR'20]



RL-based Graph2Seq for QG [Chen et al. ICLR'20]

Methods	BLEU-4	Methods	BLEU-4
$G2S_{dyn}$+BERT+RL	18.06	$G2S_{dyn}$ w/o feat	16.51
$G2S_{sta}$+BERT+RL	18.30	$G2S_{sta}$ w/o feat	16.65
$G2S_{sta}$+BERT-fixed+RL	18.20	$G2S_{dyn}$ w/o DAN	12.58
$G2S_{dyn}$+BERT	17.56	$G2S_{sta}$ w/o DAN	12.62
$G2S_{sta}$+BERT	18.02	$G2S_{sta}$ w/ DAN-word only	15.92
$G2S_{sta}$+BERT-fixed	17.86	$G2S_{sta}$ w/ DAN-contextual only	16.07
$G2S_{dyn}$+RL	17.18	$G2S_{sta}$ w/ GGNN-forward	16.53
$G2S_{sta}$+RL	17.49	$G2S_{sta}$ w/ GGNN-backward	16.75
$G2S_{dyn}$	16.81	$G2S_{sta}$ w/o BiGGNN, w/ Seq2Seq	16.14
$G2S_{sta}$	16.96	$G2S_{sta}$ w/o BiGGNN, w/ GCN	14.47

Ablation study on the SQuAD split-2 test set.

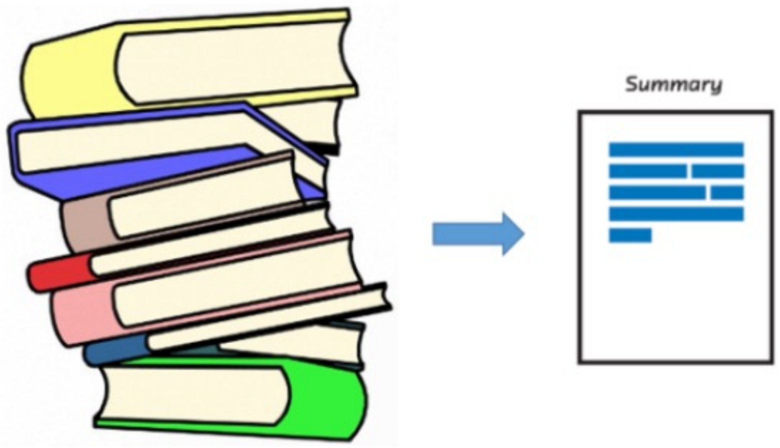
Bidirectional GNN performs better

Graph2Seq performs better than Seq2Seq

Static graph construction performs slightly better

Summarization

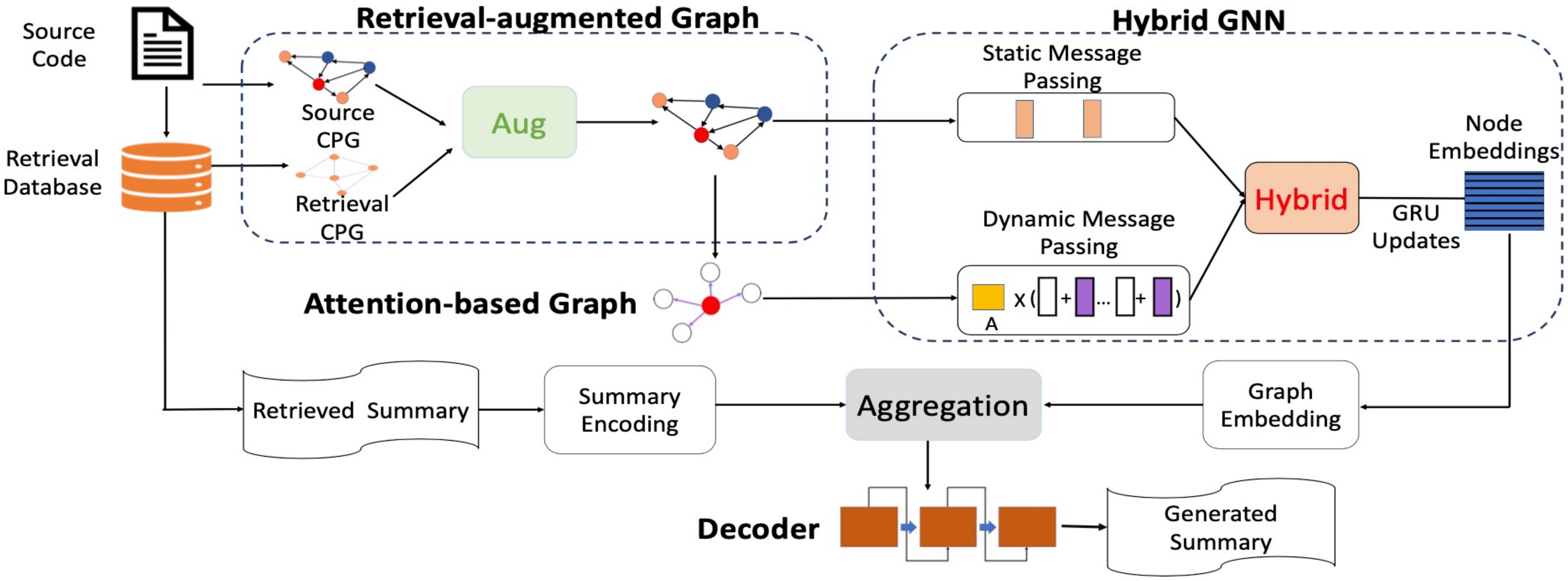
Summarization



- Input
  - A document, dialogue, code or multiple ones
- Output
  - A succinct sentence or paragraph

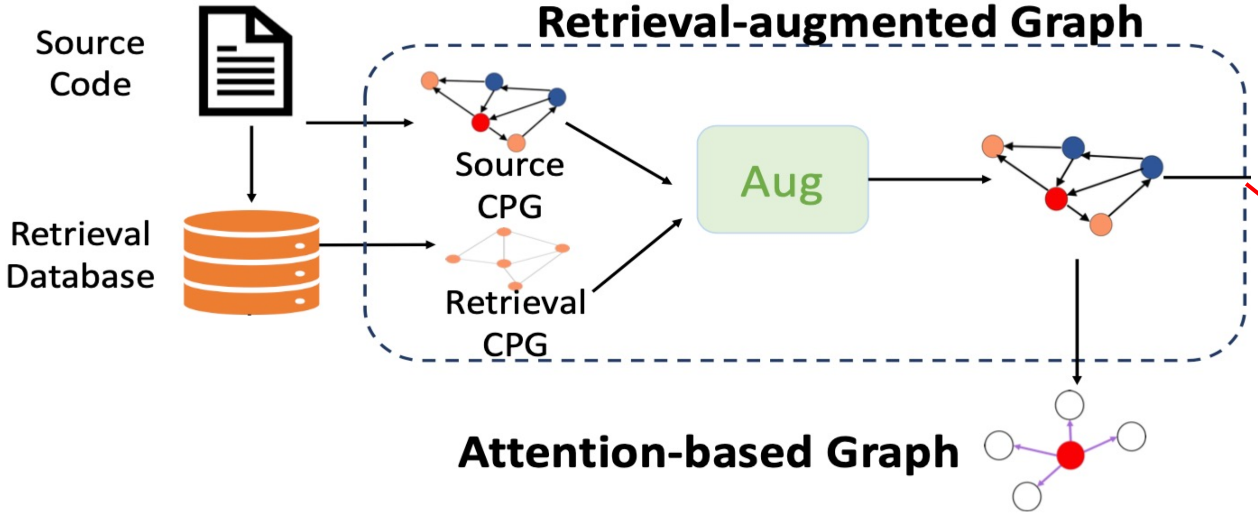
Ref: https://www.queppelin.com/how-nlp-is-helping-in-automatic-text-summarization-2, https://www.fiverr.com/yosree22/summarize-any-long-text-or-article-for-you

GNN for Code Summarization [Liu et al. ICLR'21]

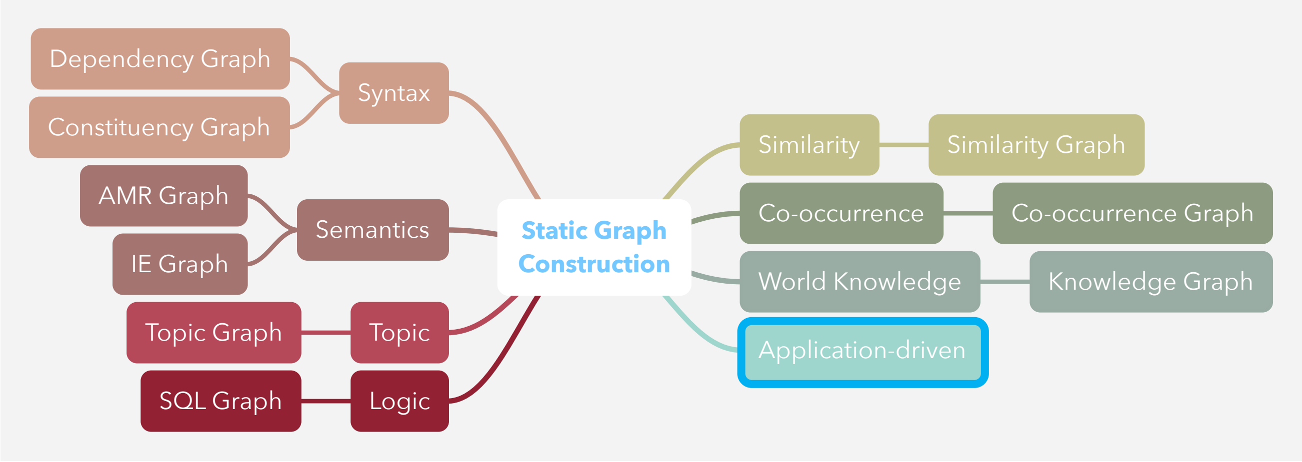


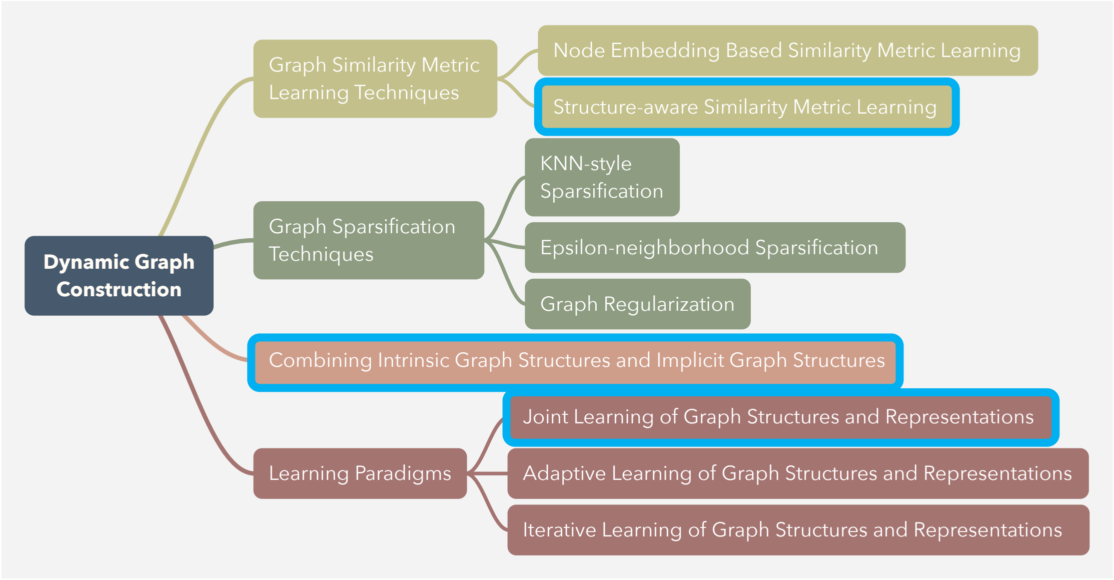
Liu et al. *"Retrieval-Augmented Generation for Code Summarization via Hybrid GNN". ICLR 2021.*

GNN for Code Summarization [Liu et al. ICLR'21]

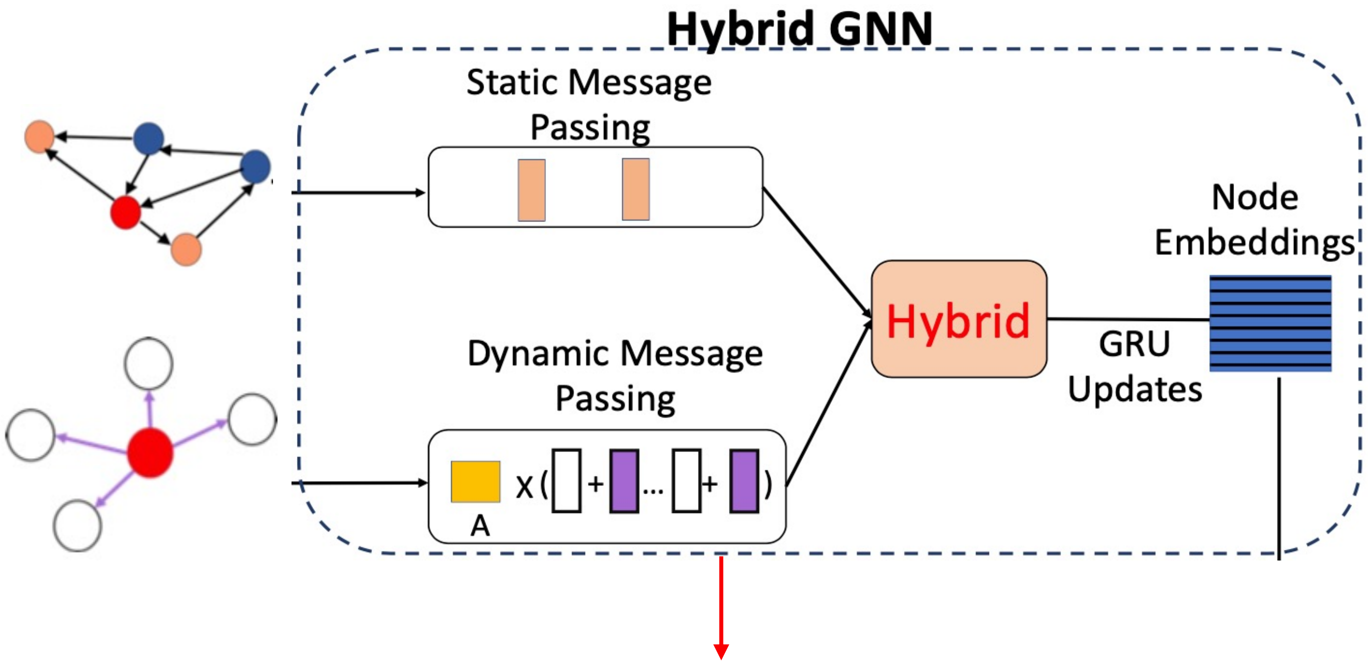


Static CPG graph + attention-based dynamic graph

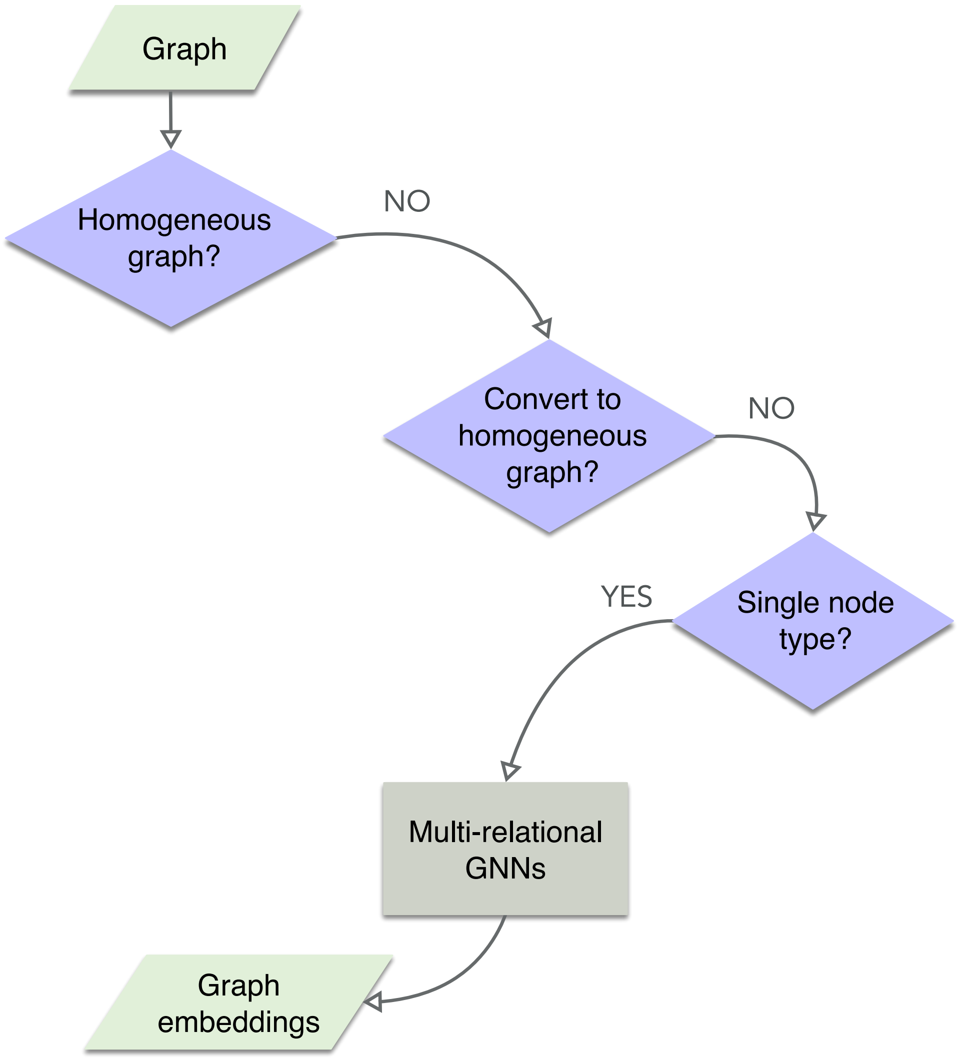




GNN for Code Summarization [Liu et al. ICLR'21]



Hybrid GNN running message passing on static & dynamic graphs



GNN for Code Summarization [Liu et al. ICLR'21]

Methods	In-domain			Out-of-domain			Overall		
	BLEU-4	ROUGE-L	METEOR	BLEU-4	ROUGE-L	METEOR	BLEU-4	ROUGE-L	METEOR
TF-IDF	15.20	27.98	13.74	5.50	15.37	6.84	12.19	23.49	11.43
NNGen	15.97	28.14	13.82	5.74	16.33	7.18	12.76	23.93	11.58
CODE-NN	10.08	26.17	11.33	3.86	15.25	6.19	8.24	22.28	9.61
Hybrid-DRL	9.29	30.00	12.47	6.30	24.19	10.30	8.42	28.64	11.73
Transformer	12.91	28.04	13.83	5.75	18.62	9.89	10.69	24.65	12.02
Dual Model	11.49	29.20	13.24	5.25	21.31	9.14	9.61	26.40	11.87
Rencos	14.80	31.41	14.64	7.54	23.12	10.35	12.59	28.45	13.21
GCN2Seq	9.79	26.59	11.65	4.06	18.96	7.76	7.91	23.67	10.23
GAT2Seq	10.52	26.17	11.88	3.80	16.94	6.73	8.29	22.63	10.00
SeqGNN	10.51	29.84	13.14	4.94	20.80	9.50	8.87	26.34	11.93
HGNN w/o augment & static	11.75	29.59	13.86	5.57	22.14	9.41	9.98	26.94	12.05
HGNN w/o augment & dynamic	11.85	29.51	13.54	5.45	21.89	9.59	9.93	26.80	12.21
HGNN w/o augment	12.33	29.99	13.78	5.45	22.07	9.46	10.26	27.17	12.32
HGNN w/o static	15.93	33.67	15.67	7.72	24.69	10.63	13.44	30.47	13.98
HGNN w/o dynamic	15.77	33.84	15.67	7.64	24.72	10.73	13.31	30.59	14.01
HGNN	**16.72**	**34.29**	**16.25**	**7.85**	**24.74**	**11.05**	**14.01**	**30.89**	**14.50**

Automatic evaluation results (in %) on the CCSD test set.

Combining static + dynamic graphs performs better

Outline

DLG4NLP Introduction

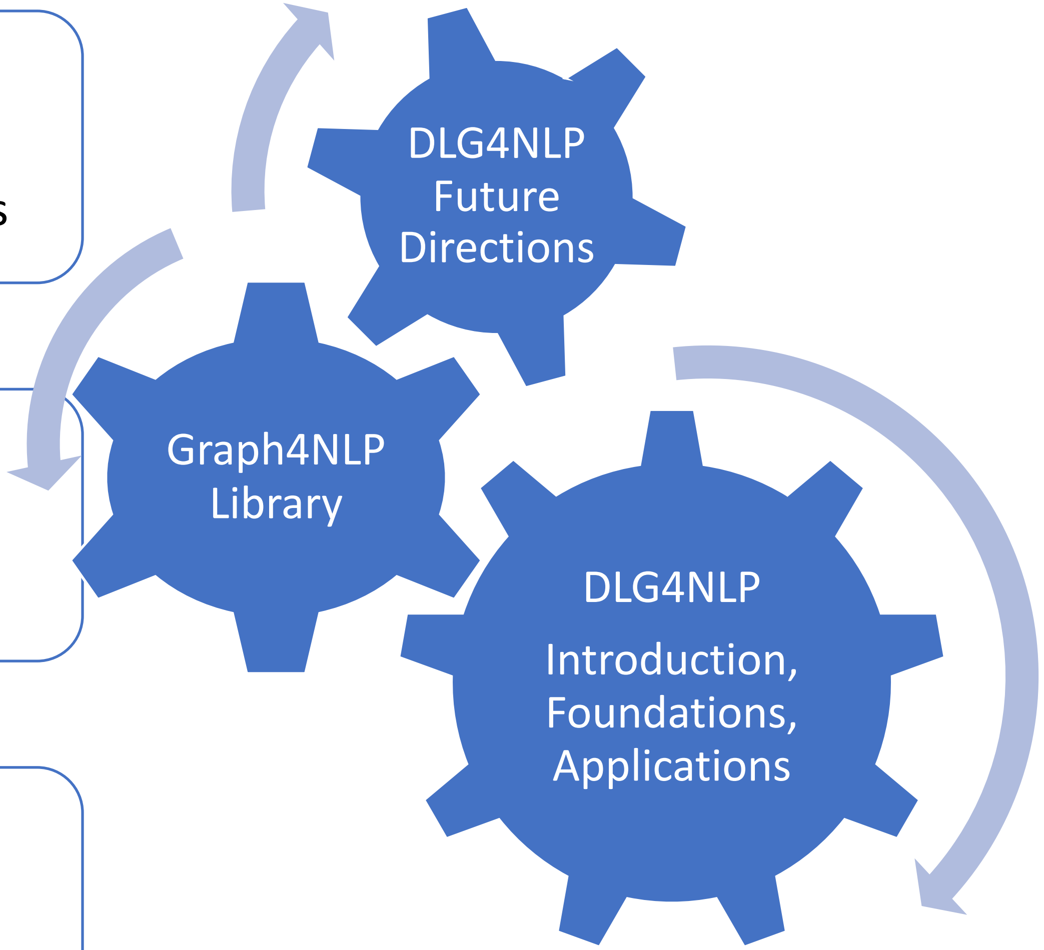
- Why Graphs for NLP?
- Conventional ML for NLP
- Deep Learning on Graphs: Foundations and Models

DLG4NLP Foundations

- Graph Construction for NLP
- Graph Representation Learning for NLP
- Graph Encoder-Decoder Models for NLP

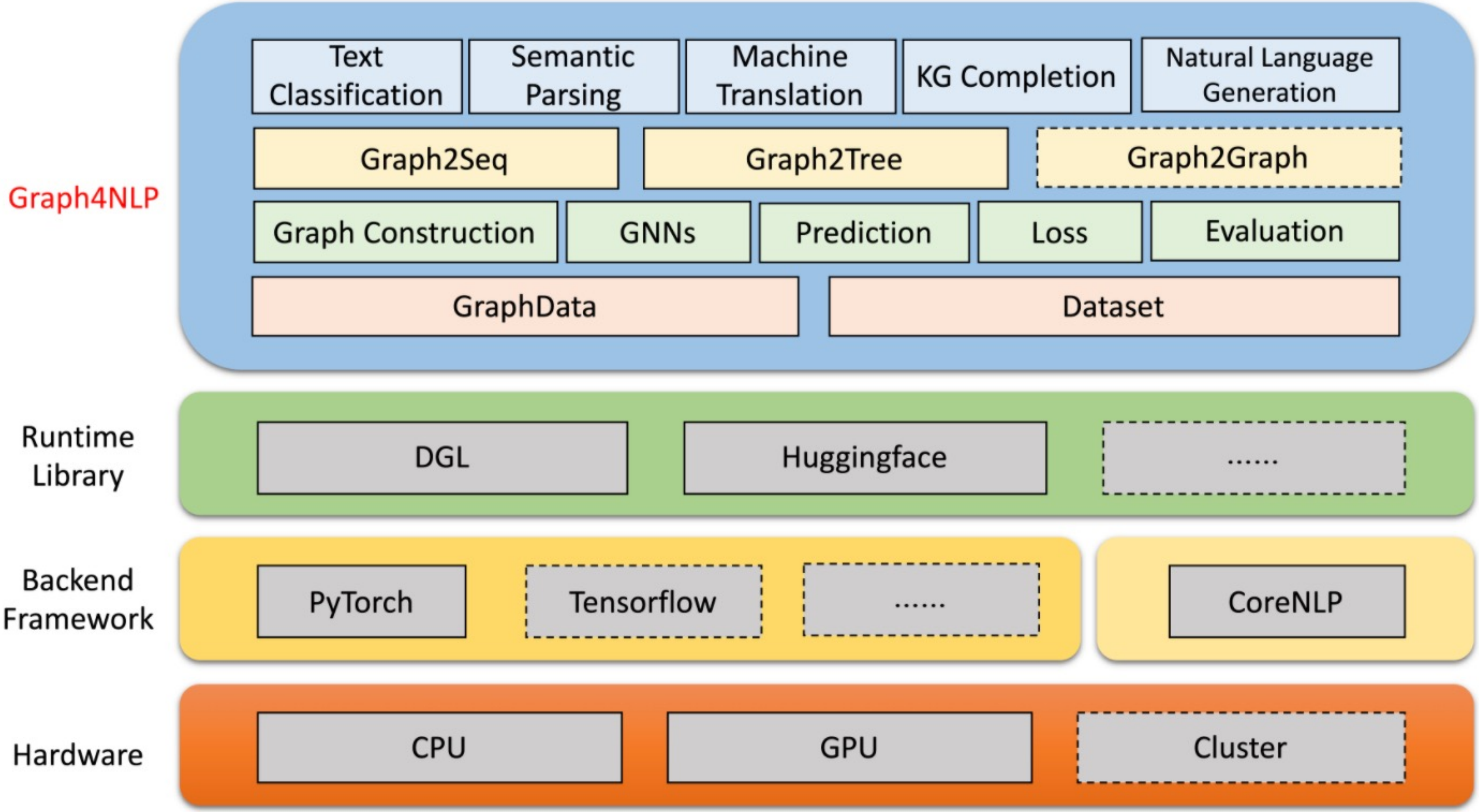
DLG4NLP Applications

- Natural Question Generation
- Summarization



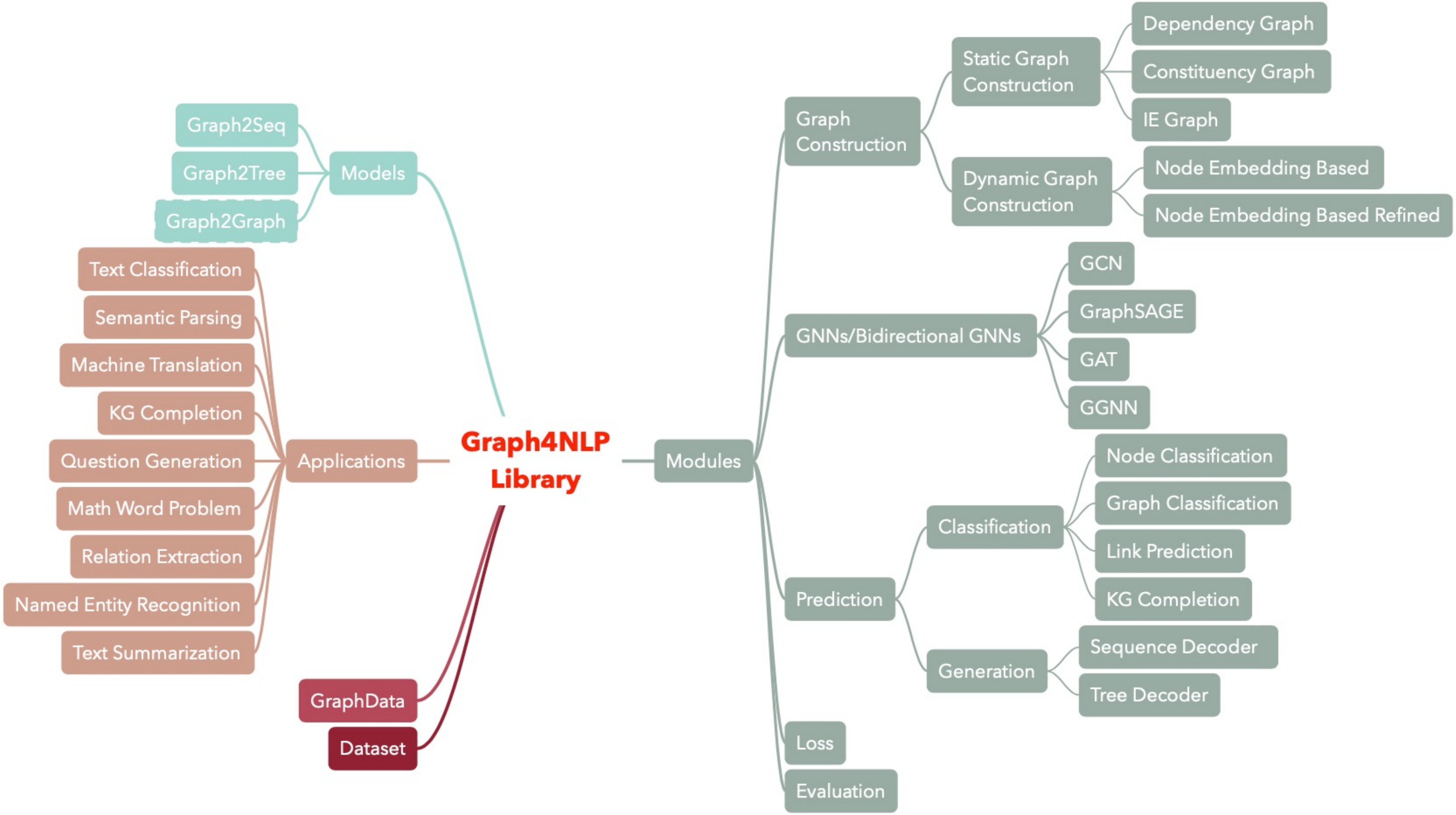
Graph4NLP: A Library for Deep Learning on Graphs for NLP

Overall Architecture of Graph4NLP Library

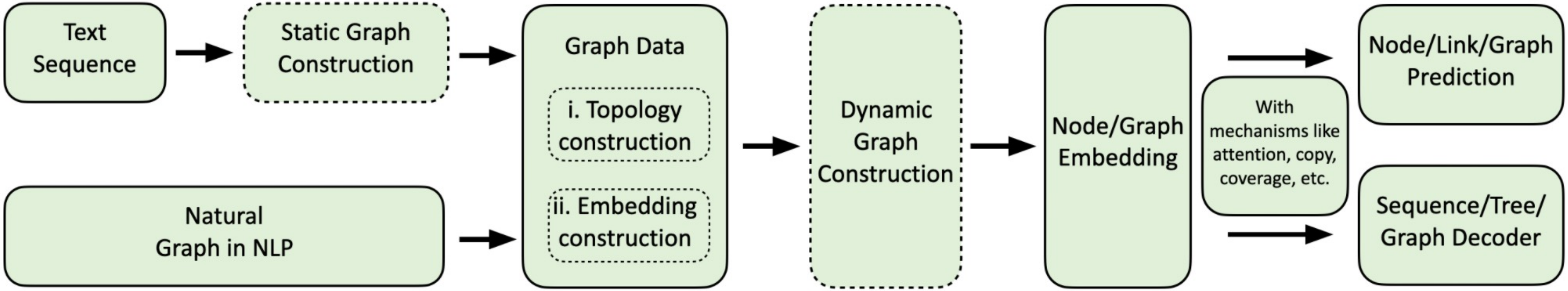


DGL: https://github.com/dmlc/dgl, DIG: https://github.com/divelab/DIG, Huggingface: https://github.com/huggingface/transformers

Dive Into Graph4NLP Library



Computation Flow of Graph4NLP



Performance of Built-in NLP Tasks

Task	Dataset	GNN Model	Graph construction	Evaluation	Performance
Text classification	TRECT CAirline CNSST	GAT	Dependency	Accuracy	0.948 0.769 0.538
Semantic Parsing	JOBS	SAGE	Constituency	Execution accuracy	0.936
Question generation	SQuAD	GGNN	Dependency	BLEU-4	0.15175
Machine translation	IWSLT14	GCN	Dynamic	BLEU-4	0.3212
Summarization	CNN(30k)	GCN	Dependency	ROUGE-1	26.4
Knowledge graph completion	Kinship	GCN	Dependency	MRR	82.4
Math word problem	MAWPS MATHQA	SAGE	Dynamic	Solution accuracy Exact match	76.4 61.07

Ref: https://mentorphile.com/2018/09/14/demo-or-die/

Demo 1: Building a Text Classification Application

1) git clone https://github.com/graph4ai/graph4nlp_demo
2) follow Get Started instructions in README

Demo 1: Building a Text Classification Application

```
def forward(self, graph_list, tgt=None, require_loss=True):
    # build graph topology
    batch_gd = self.graph_topology(graph_list)

    # run GNN encoder
    self.gnn(batch_gd)

    # run graph classifier
    self.clf(batch_gd)
    logits = batch_gd.graph_attributes['logits']

    if require_loss:
        loss = self.loss(logits, tgt)
        return logits, loss
    else:
        return logits
```

Model arch

https://github.com/graph4ai/graph4nlp_demo/tree/main/NAACL2021_demo

Demo 1: Building a Text Classification Application

Graph construction API, various built-in options, can be customized

```
self.graph_topology = DependencyBasedGraphConstruction(
                        embedding_style=embedding_style,
                        vocab=vocab.in_word_vocab,
                        hidden_size=config['num_hidden'],
                        word_dropout=config['word_dropout'],
                        rnn_dropout=config['rnn_dropout'],
                        fix_word_emb=not config['no_fix_word_emb'],
                        fix_bert_emb=not config.get('no_fix_bert_emb', False))
```

https://github.com/graph4ai/graph4nlp_demo/tree/main/NAACL2021_demo

Demo 1: Building a Text Classification Application

GNN API, various built-in options, can be customized

```
self.gnn = GraphSAGE(config['gnn_num_layers'],
            config['num_hidden'],
            config['num_hidden'],
            config['num_hidden'],
            config['graphsage_aggreagte_type'],
            direction_option=config['gnn_direction_option'],
            feat_drop=config['gnn_dropout'],
            bias=True,
            norm=None,
            activation=F.relu,
            use_edge_weight=use_edge_weight)
```

https://github.com/graph4ai/graph4nlp_demo/tree/main/NAACL2021_demo

Demo 1: Building a Text Classification Application

Prediction API, various built-in options, can be customized

```
self.clf = FeedForwardNN(2 * config['num_hidden'] \
                if config['gnn_direction_option'] == 'bi_sep' \
                else config['num_hidden'],
                config['num_classes'],
                [config['num_hidden']],
                graph_pool_type=config['graph_pooling'],
                dim=config['num_hidden'],
                use_linear_proj=config['max_pool_linear_proj'])
```

https://github.com/graph4ai/graph4nlp_demo/tree/main/NAACL2021_demo

Graph4NLP

Demo 1: Building a Text Classification Application

Dataset API, various built-in options, can be customized

```
dataset = TrecDataset(root_dir=self.config.get('root_dir', self.config['root_data_dir']),
                      pretrained_word_emb_name=self.config.get('pretrained_word_emb_name', "840B"),
                      merge_strategy=merge_strategy,
                      seed=self.config['seed'],
                      thread_number=4,
                      port=9000,
                      timeout=15000,
                      word_emb_size=300,
                      graph_type=graph_type,
                      topology_builder=topology_builder,
                      topology_subdir=topology_subdir,
                      dynamic_graph_type=self.config['graph_type'] if \
                          self.config['graph_type'] in ('node_emb', 'node_emb_refined') else None,
                      dynamic_init_topology_builder=dynamic_init_topology_builder,
                      dynamic_init_topology_aux_args={'dummy_param': 0})
```

https://github.com/graph4ai/graph4nlp_demo/tree/main/NAACL2021_demo

Outline

Graph4NLP

DLG4NLP Introduction

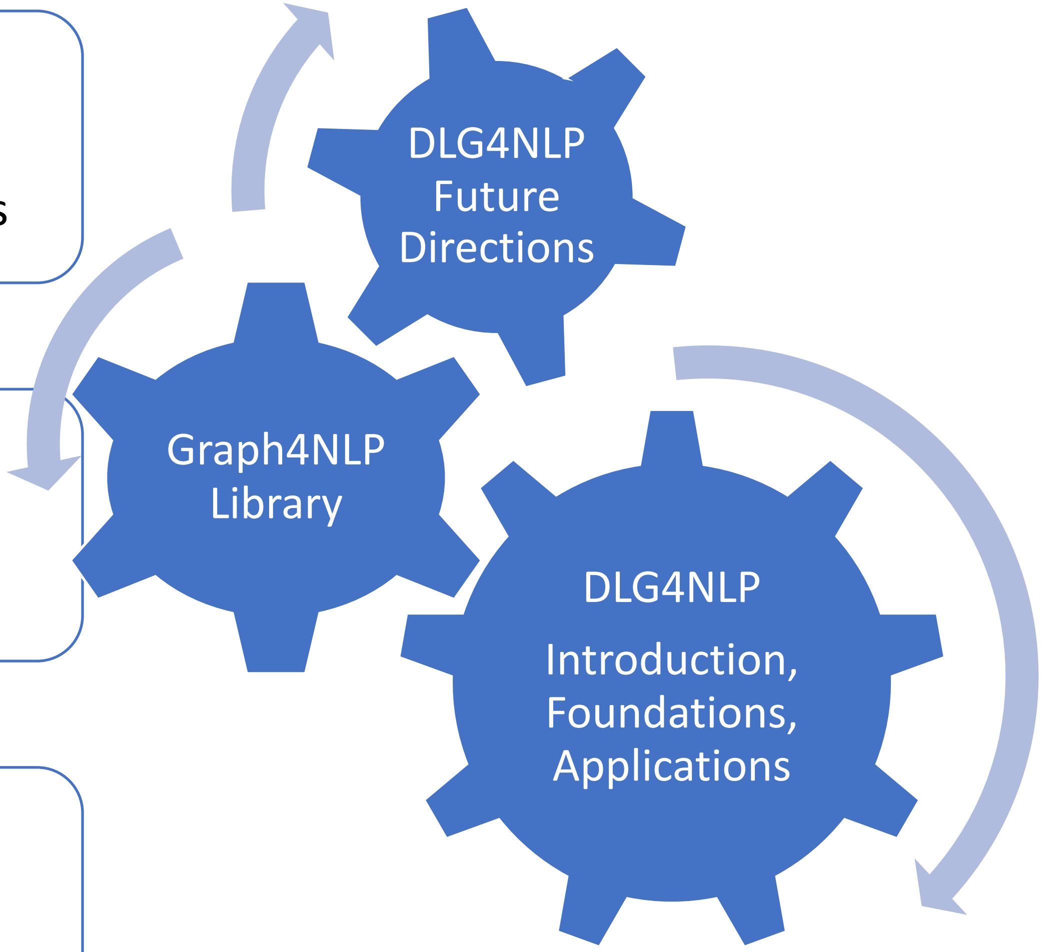
- Why Graphs for NLP?
- Conventional ML for NLP
- Deep Learning on Graphs: Foundations and Models

DLG4NLP Foundations

- Graph Construction for NLP
- Graph Representation Learning for NLP
- Graph Encoder-Decoder Models for NLP

DLG4NLP Applications

- Natural Question Generation
- Summarization



DLG4NLP: Future Directions and Conclusions

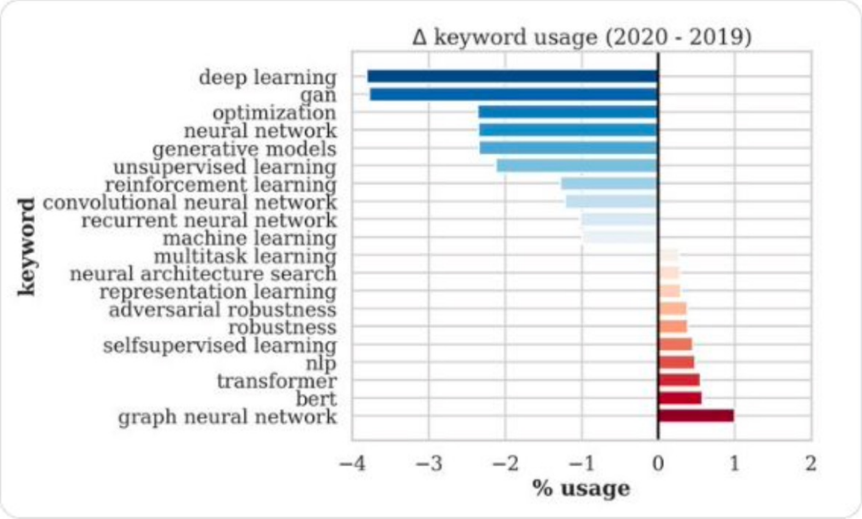
Graph4NLP

Future Directions

[Vashishth et al. EMNLP’19 Tutorial]

- The Rise of GNN + NLP

#ICLR2020 submissions on graph neural networks, NLP and robustness have the greatest growth. @iclr_conf @openreviewnet



- Graph Construction for NLP
  - Dynamic graph construction are largely underexplored!
  - How to effectively combine advantages of static graph and dynamic graph?
  - How to construct heterogeneous dynamic graph?
  - How to make dynamic graph construction itself scalable?

Future Directions

- Scaling GNNs to Large Graphs
  - Most existing multi-relational or heterogeneous GNNs will have scalability issues when applied to large graphs in NLP such as KGs (> 1m)
- GNNs + Transformer in NLP
  - How to effectively combine the advantages of GNNs and Transformer?
  - Is graph transformer the best way to utilize?
- Pretraining GNNs for NLP
  - Information Retrieval/ Search

Future Directions

- Graph-to-graph Learning in NLP
  - How to effectively develop Graph-to-Graph models for solving graph transformation problem in NLP (i.e. information extraction)?
- Joint Text and KG Reasoning in NLP
  - Joint text and KG reasoning is less explored although GNNs for multi-hop reasoning gains popularity
- Incorporate Source and Context into Knowledge Graph Construction and Verification

Conclusions

- Deep Learning on Graphs for NLP is a fast-growing area today!
- Since graph can naturally encode complex information, it could bridge a gap by combining both empirical domain knowledges and the power of deep learning.
- For a NLP task,
    - how to convert text sequence into the best graph (directed, multi-relation, heterogeneous)
    - how to determine proper graph representation learning technique?
- Our Graph4NLP library aims to make easy use of GNNs for NLP:
    - Code: https://github.com/graph4ai/graph4nlp
    - Demo: https://github.com/graph4ai/graph4nlp_demo
    - Github literature list: https://github.com/graph4ai/graph4nlp_literature
- GNN4NLP survey: https://arxiv.org/pdf/2106.06090