GCN for Dependency Parsing

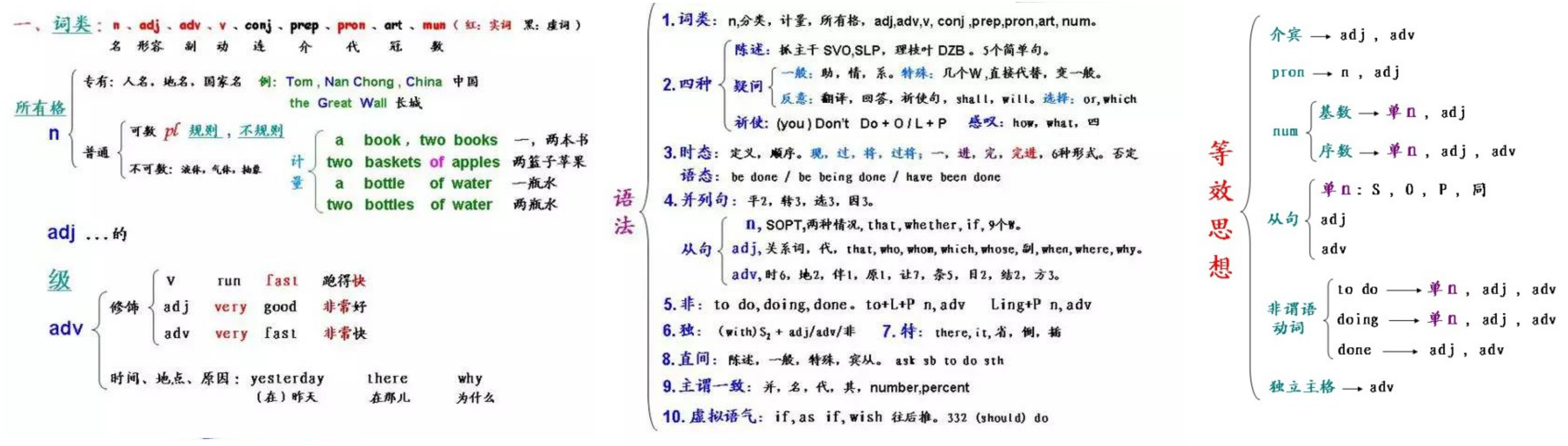
November 6th 2019

Outline

1. Background
2. Task Definition
3. Graph-based Dependency Parsing with Graph Neural Networks

1. Syntax

- The word **syntax** comes from the Greek syntaxis , meaning “setting out together or arrangement”, and refers to **the way words are arranged together**.



Jurafsky, Daniel, and James H. Martin. "Speech and Language Processing: An introduction to speech recognition, computational linguistics and natural language processing." *Upper Saddle River, NJ: Prentice Hall* (2008).
http://www.sohu.com/a/74710386_356310

1.1 Part-Of-Speech

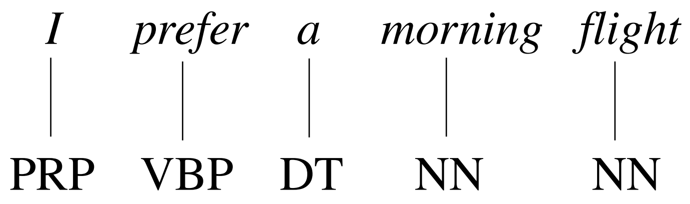
- Parts of speech (POS): word classes or syntactic categories
  - noun, verb, pronoun, preposition, adverb, conjunction, participle, and article
- Example: The Penn TreeBank

Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coordinating conjunction	*and, but, or*	PDT	predeterminer	*all, both*	VBP	verb non-3sg present	*eat*
CD	cardinal number	*one, two*	POS	possessive ending	*'s*	VBZ	verb 3sg pres	*eats*
DT	determiner	*a, the*	PRP	personal pronoun	*I, you, he*	WDT	wh-determ.	*which, that*
EX	existential 'there'	*there*	PRP$	possess. pronoun	*your, one's*	WP	wh-pronoun	*what, who*
FW	foreign word	*mea culpa*	RB	adverb	*quickly*	WP$	wh-possess.	*whose*
IN	preposition/ subordin-conj	*of, in, by*	RBR	comparative adverb	*faster*	WRB	wh-adverb	*how, where*
JJ	adjective	*yellow*	RBS	superlatv. adverb	*fastest*	$	dollar sign	$
JJR	comparative adj	*bigger*	RP	particle	*up, off*	#	pound sign	#
JJS	superlative adj	*wildest*	SYM	symbol	+,%, &	"	left quote	' or "
LS	list item marker	*1, 2, One*	TO	"to"	*to*	"	right quote	' or "
MD	modal	*can, should*	UH	interjection	*ah, oops*	(	left paren	[, (, {, <
NN	sing or mass noun	*llama*	VB	verb base form	*eat*	)	right paren	],), }, >
NNS	noun, plural	*llamas*	VBD	verb past tense	*ate*	,	comma	,
NNP	proper noun, sing.	*IBM*	VBG	verb gerund	*eating*	.	sent-end punc	. ! ?
NNPS	proper noun, plu.	*Carolinas*	VBN	verb past part.	*eaten*	:	sent-mid punc	: ; ... – -

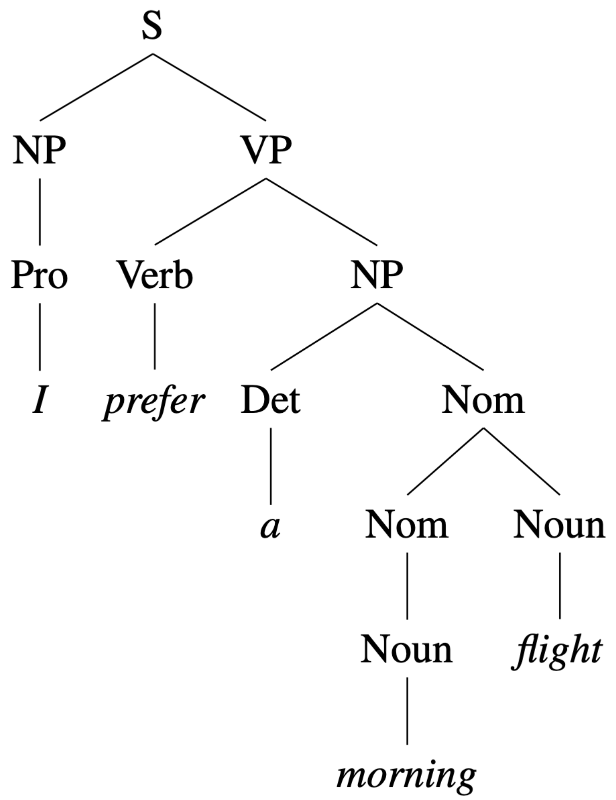
Marcus, Mitchell, et al. "The Penn Treebank: annotating predicate argument structure." *Proceedings of the workshop on Human Language Technology*. Association for Computational Linguistics, 1994.

1.2 Constituency Tree

- Constituency: abstraction for groups of words -> hierarchical graph.



PRP: Personal Pronoun
VBP: Verb Present
DT: Determiner
NN: Single or Mass Noun

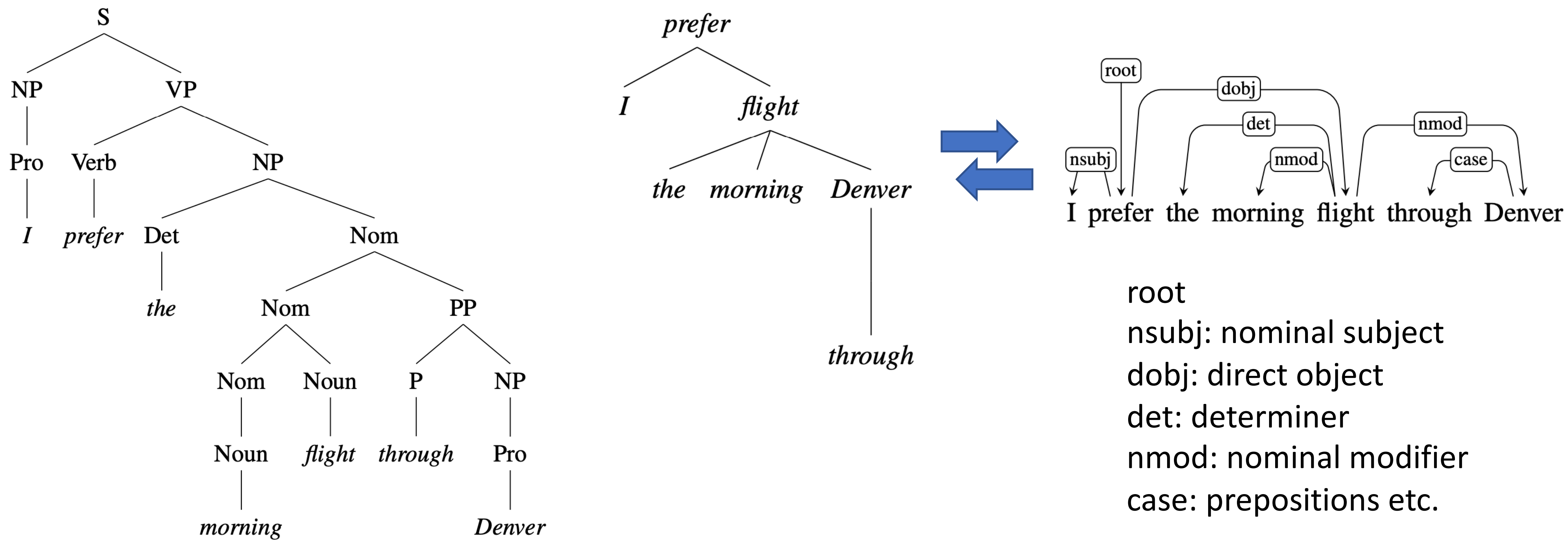


S: Sentence
NP: Noun Phrase
VP: Verb Phrase
Pro: Pronoun
Det: Determiner
Nom: Nominal Noun

Jurafsky, Daniel, and James H. Martin. "Speech and Language Processing: An introduction to speech recognition, computational linguistics and natural language processing." *Upper Saddle River, NJ: Prentice Hall* (2008).

1.3 Dependency Tree

- Dependency: relation between a pair of words -> dependency tree.



root
nsubj: nominal subject
dobj: direct object
det: determiner
nmod: nominal modifier
case: prepositions etc.

Important property: DAG constrain
(Directed Acyclic Graph)

Jurafsky, Daniel, and James H. Martin. "Speech and Language Processing: An introduction to speech recognition, computational linguistics and natural language processing." *Upper Saddle River, NJ: Prentice Hall* (2008).

2. Dependency Parsing via Graph Algorithm

- Given

$\mathrm{s} = w_1, \dots, w_n$	A sentence containing n words
$\mathrm{G} = (V, E)$	The complete graph over the sentence s.
V	The set of nodes ($w_1, \dots, w_n$ and a root node)
E	The set of node edges $\mathrm{e} = (i, j, r)$, where $w_i \xrightarrow{r} w_j$

- Find the best dependency tree $\hat{T}$ which has the highest score.

$$\hat{T} = \underset{T \in G}{\operatorname{argmax}}\, score(T, s)$$

$$score(T, s) = \sum_{e \in T} score(e, s)$$

2. Dependency Parsing via Graph Algorithm

- Two steps for solving the problem:
    1) Compute scores for each edge. (*main focus of the paper)
    2) Infer dependency tree based on the estimated edge scores.
        i. Search for optimal tree (without edge labels)
        ii. Predict edge labels

3. The paper

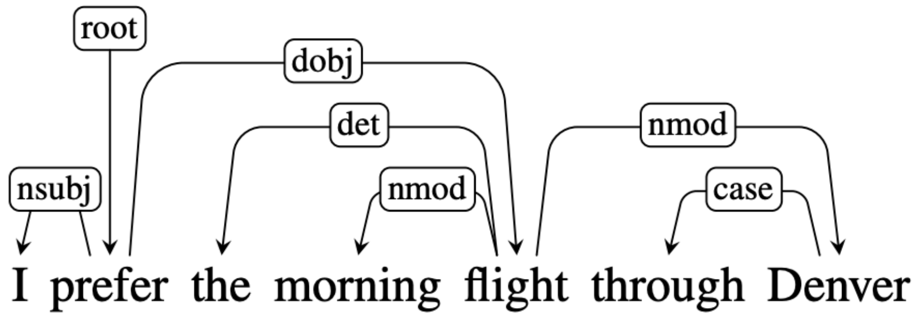
Graph-based Dependency Parsing with Graph Neural Networks

Tao Ji , Yuanbin Wu , and Man Lan

Department of Computer Science and Technology,
East China Normal University
{taoji.cs}@gmail.com {ybwu,mlan}@cs.ecnu.edu.cn

3.1 Motivation

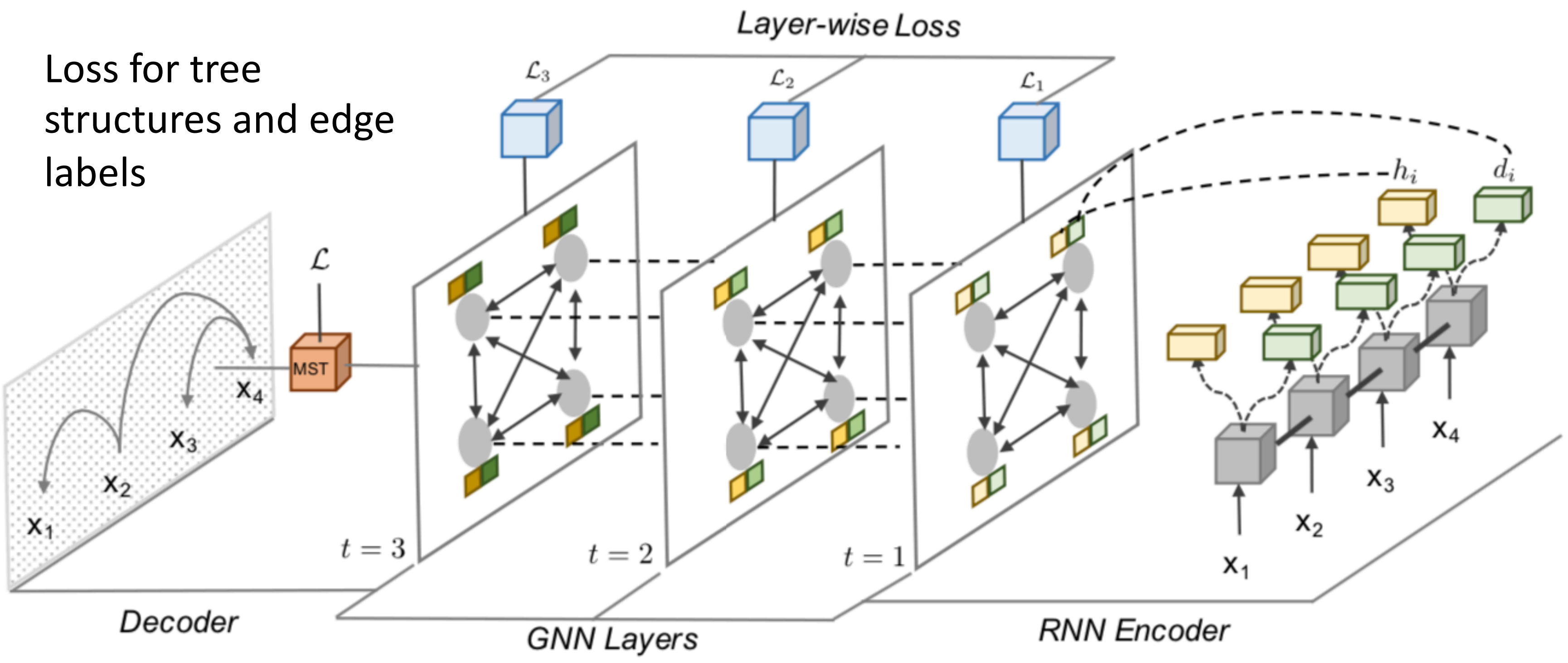
- Previous work has shown first-order relations can help learning.
  - Head and dependent vector.
- Introduce structure knowledge into node representations.
  - Especially high-order relations (e.g. grandparents)
  - Graph neural network to capture long-term dependencies.



3.2 Overview

Loss for predicting edge scores

Loss for tree structures and edge labels



Infer optimal tree given a complete graph with edge scores.

Encode high-order information and predict edge scores

Basic node representations

3.3 RNN Encoder

$\sigma(i,j)$	Score for edge $w_i \rightarrow w_j$
W, A, b_*	Weights
$\bigoplus$	Concatenation operation
$e(pos_i)$	Part of speech embedding
$e'(w_i)$	Randomly initialized word embedding
$e(w_i)$	Pretrained word embedding
θ	Parameters

$$\sigma(i,j) = \mathrm{Softmax}_i\left(\boldsymbol{h}_i^\intercal A \boldsymbol{d}_j + \boldsymbol{b}_1^\intercal \boldsymbol{h}_i + \boldsymbol{b}_2^\intercal \boldsymbol{d}_j\right)$$

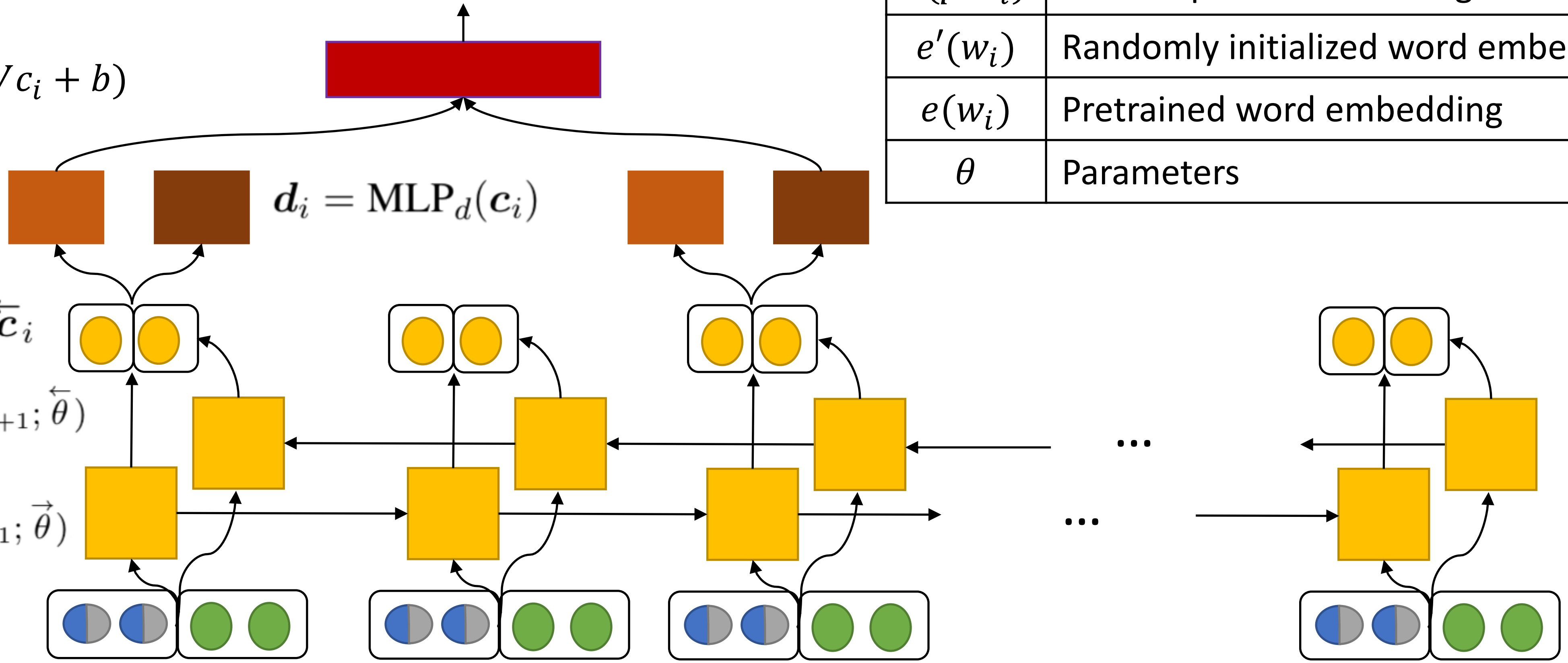
$$MLP = ReLu(Wc_i + b)$$

$$\boldsymbol{h}_i = \mathrm{MLP}_h(\boldsymbol{c}_i) \qquad \boldsymbol{d}_i = \mathrm{MLP}_d(\boldsymbol{c}_i)$$

$$\boldsymbol{c}_i = \vec{\boldsymbol{c}}_i \oplus \overleftarrow{\boldsymbol{c}}_i$$

$$\overleftarrow{\boldsymbol{c}}_i = \mathrm{LSTM}(\boldsymbol{x}_i, \overleftarrow{\boldsymbol{c}}_{i+1}; \overleftarrow{\theta})$$

$$\vec{\boldsymbol{c}}_i = \mathrm{LSTM}(\boldsymbol{x}_i, \vec{\boldsymbol{c}}_{i-1}; \vec{\theta})$$

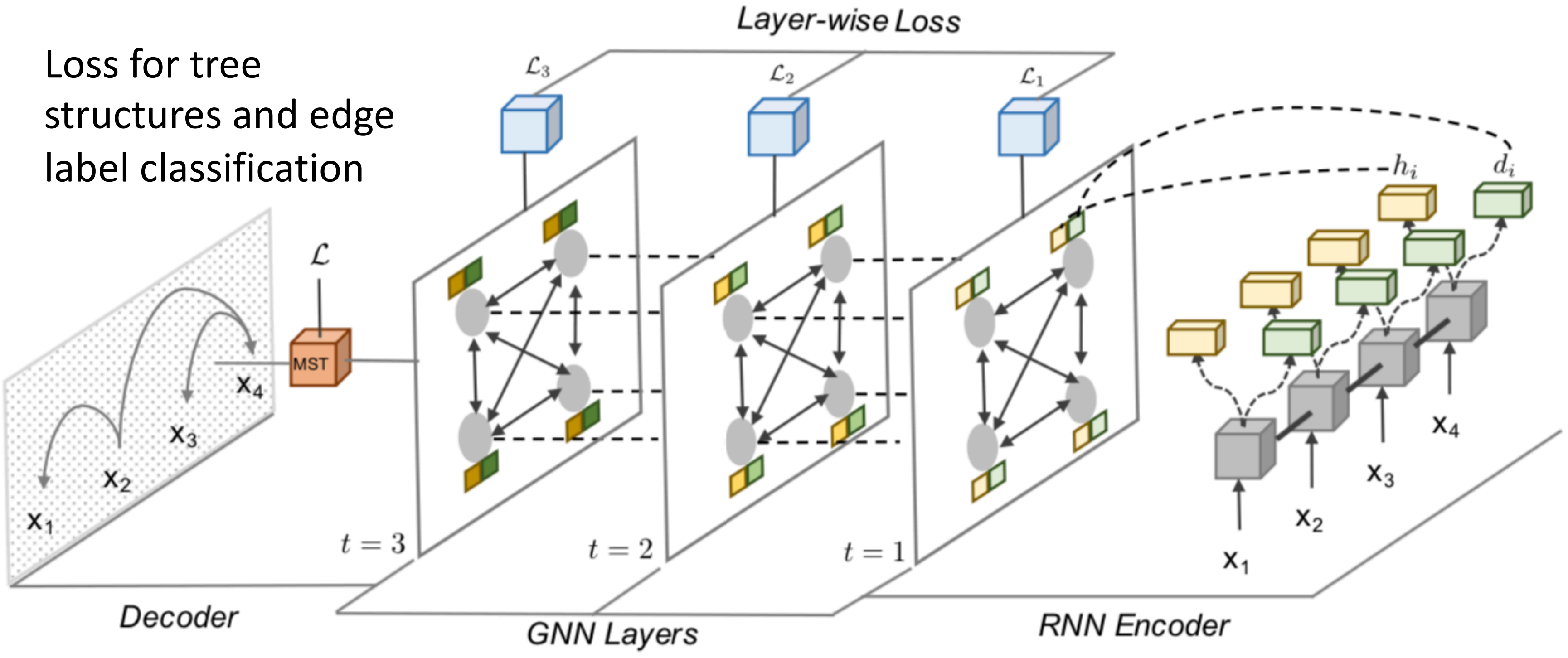


$$\boldsymbol{x}_i = \left(\boldsymbol{e}(w_i) + \boldsymbol{e}'(w_i)\right) \oplus \boldsymbol{e}(pos_i)$$

Overview

Loss for predicting edge scores

Loss for tree structures and edge label classification



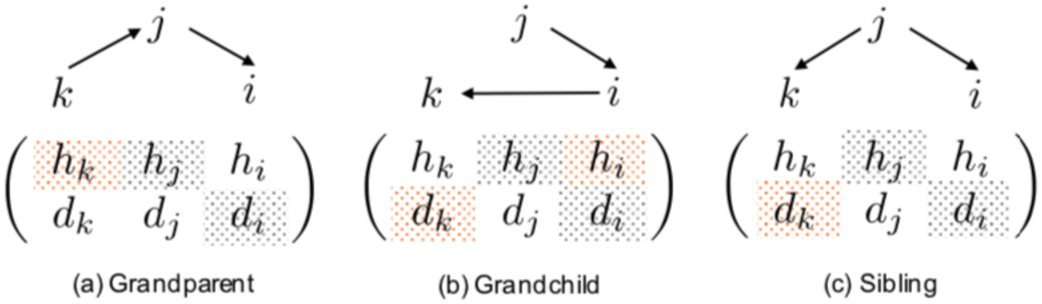
Infer optimal tree given a complete graph with edge scores.

Encoding high-order information

Basic node representations

3.4 GNN Layers

- Encode three types of high-order information for edge (j, i)

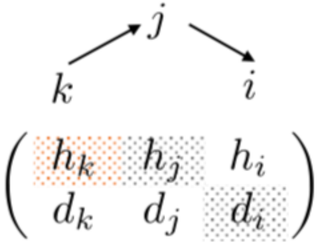


1) The grey shadows indicate first-order information: parent -> children
2) The orange shadows are the second-order information.

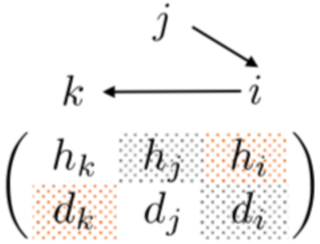
- Multi-layer GCN is able to explicitly capture this high-order information.

3.4 GNN Layers (cont')

- Encode three types of high-order information for edge (j, i)

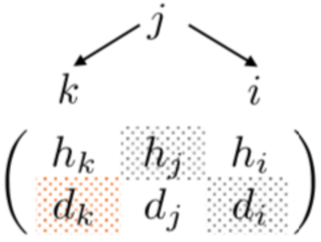


(a) Grandparent



(b) Grandchild

$$
\begin{cases}
\boldsymbol{h}_i^t = g\Big(W_1 \sum_{j \in \mathcal{N}(i)} \alpha_{ji}^t \boldsymbol{h}_j^{t-1} + B_1 \boldsymbol{h}_i^{t-1}\Big) \\
\boldsymbol{d}_i^t = g\Big(W_2 \sum_{j \in \mathcal{N}(i)} \alpha_{ij}^t \boldsymbol{d}_j^{t-1} + B_2 \boldsymbol{d}_i^{t-1}\Big).
\end{cases}
$$



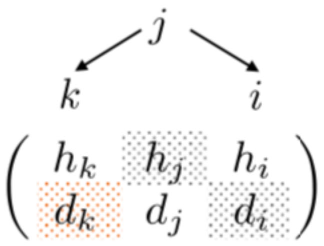
(c) Sibling

$$
\begin{cases}
\boldsymbol{h}_i^t = g\Big(W_1 \sum_{j \in \mathcal{N}(i)} \alpha_{ij}^t \boldsymbol{d}_j^{t-1} + B_1 \boldsymbol{h}_i^{t-1}\Big) \\
\boldsymbol{d}_i^t = g\Big(W_2 \sum_{j \in \mathcal{N}(i)} \alpha_{ji}^t \boldsymbol{h}_j^{t-1} + B_2 \boldsymbol{d}_i^{t-1}\Big).
\end{cases}
$$

W_1, B_1	Weights for updating the head vector
W_2, B_2	Weights for updating the dependent vector
α_{ij}^t	Score for edge (i, j) at the t^{th} layer.
h_i^t	The head vector for node i at the t^{th} layer.
d_i^t	The dependent vector for node i at the t^{th} layer.
g	Non-linear activation function

3.4 GNN Layers – My understanding

- How to gather information of neighbors?
    1) Head nodes: gather information from neighbors' dependent nodes.
    2) Dependent nodes: gather information from neighbors' head nodes.

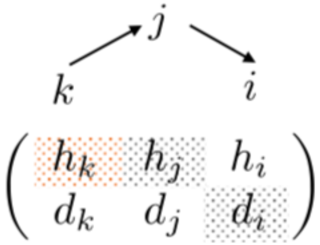


$$h_j^t = g(W_1 \sum_{i \in \mathcal{N}(j)} \alpha_{ji}^t d_i^{t-1} + B_1 h_j^{t-1})$$

$$d_i^t = g(W_2 \sum_{j \in \mathcal{N}(i)} \alpha_{ji}^t h_j^{t-1} + B_2 d_i^{t-1})$$

3.4 GNN Layers – My understanding (cont')

- Now, we want to capture the 2nd order information, how to propagate h_k to d_i via node h_j?



$$\begin{pmatrix} h_k & h_j & h_i \\ d_k & d_j & d_i \end{pmatrix}$$

(a) Grandparent

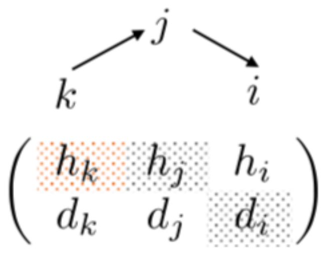
1) $h_k \rightarrow d_m$
2) $d_m \rightarrow h_j$
3) $h_j \rightarrow d_i$ for some word m

Conclusion: unable to directly capture the 2nd order information.

3.4 GNN Layers – My understanding (cont')

- How to solve this problem?
  - Shortcut from h_k to h_j.

1) h_k to h_j. (shortcut update)
2) h_j to d_i. (standard update)



(a) Grandparent

3.4 GNN Layers – Put updating functions together

- Synchronous updates:
  - update h_i and d_i at the same time

$$\begin{cases} \boldsymbol{h}_i^t = g\Big(W_1 \sum_{j \in \mathcal{N}(i)} (\alpha_{ji}^t \boldsymbol{h}_j^{t-1} + \alpha_{ij}^t \boldsymbol{d}_j^{t-1}) + B_1 \boldsymbol{h}_i^{t-1}\Big) \\ \boldsymbol{d}_i^t = g\Big(W_2 \sum_{j \in \mathcal{N}(i)} (\alpha_{ij}^t \boldsymbol{h}_j^{t-1} + \alpha_{ji}^t \boldsymbol{d}_j^{t-1}) + B_2 \boldsymbol{d}_i^{t-1}\Big). \end{cases}$$

- Asynchronous updates:
  - updates h_i first, then update d_i

$$\begin{cases} \boldsymbol{h}_i^{t-\frac{1}{2}} = g\Big(W_1 \sum_{j \in \mathcal{N}(i)} (\alpha_{ji}^t \boldsymbol{h}_j^{t-1} + \alpha_{ij}^t \boldsymbol{d}_j^{t-1}) + B_1 \boldsymbol{h}_i^{t-1}\Big) \\ \boldsymbol{d}_i^t = g\Big(W_2 \sum_{j \in \mathcal{N}(i)} (\alpha_{ij}^t \boldsymbol{h}_j^{t-\frac{1}{2}} + \alpha_{ji}^t \boldsymbol{d}_j^{t-1}) + B_2 \boldsymbol{d}_i^{t-1}\Big), \end{cases}$$

3.4 GNN Layers – Edge scores

1. Soft scores:

$$\alpha_{ij}^t = \sigma^t(i,j) = P^t(i|j)$$

$$\sigma(i,j) = \mathrm{Softmax}_i\left(\boldsymbol{h}_i^\intercal A\boldsymbol{d}_j + \boldsymbol{b}_1^\intercal \boldsymbol{h}_i + \boldsymbol{b}_2^\intercal \boldsymbol{d}_j\right)$$

2. {0, 1} scores:

$$\alpha_{ij}^t = \begin{cases} 1, & i = \arg\max_{i'} P^t(i'|j) \\ 0, & \text{otherwise} \end{cases}$$

3. Top k scores:

$$\alpha_{ij}^t = \begin{cases} \mathrm{Softmax}_i\left(\boldsymbol{h}_i^\intercal A\boldsymbol{d}_j + \boldsymbol{b}_1^\intercal \boldsymbol{h}_i + \boldsymbol{b}_2^\intercal \boldsymbol{d}_j\right), i \in \mathcal{N}_k^t(j) \\ 0, \quad \text{otherwise} \end{cases}$$

$\mathcal{N}_k(j)$ top k neighbors with highest scores

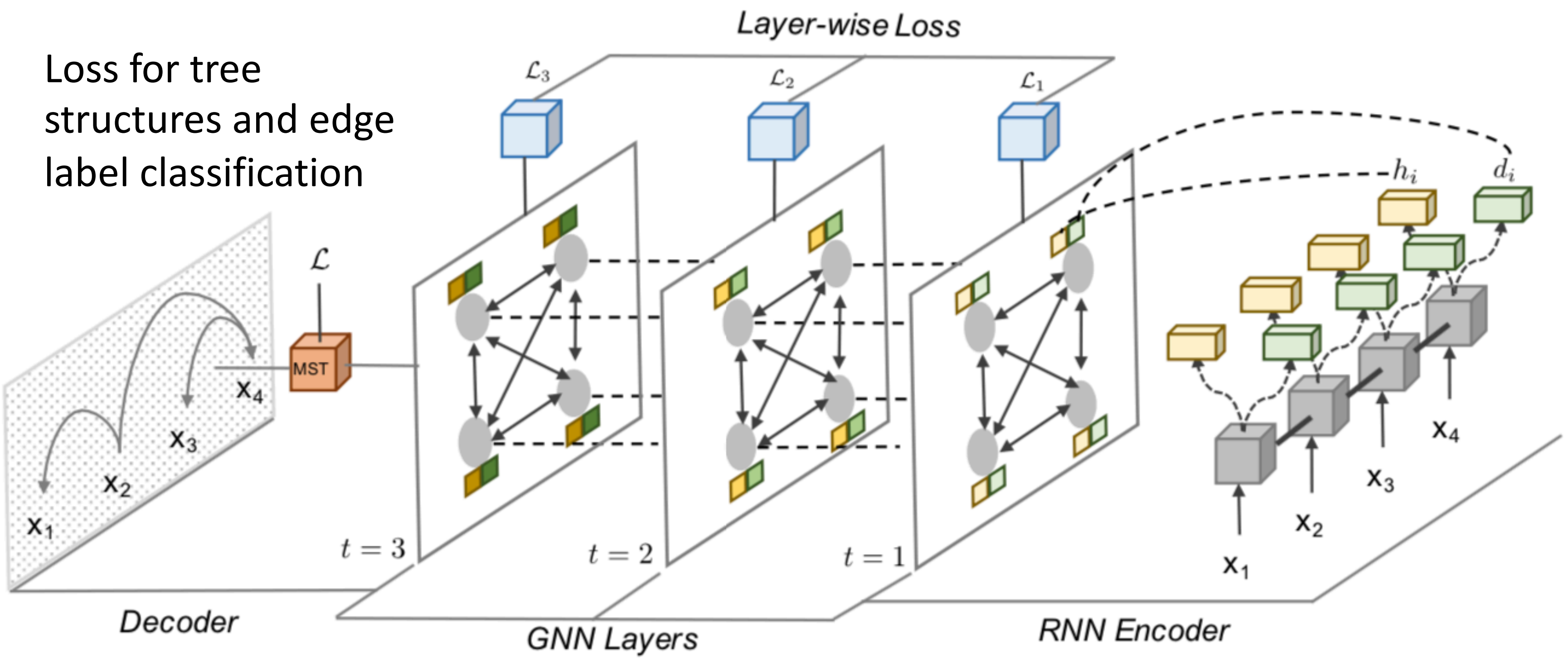
4. Average:

$$\alpha_{ij}^t = \frac{1}{n}, \quad \forall j \in V, i \in V/\{j\}$$

Overview

Loss for predicting edge scores

Loss for tree structures and edge label classification



Infer optimal tree given a complete graph with edge scores.

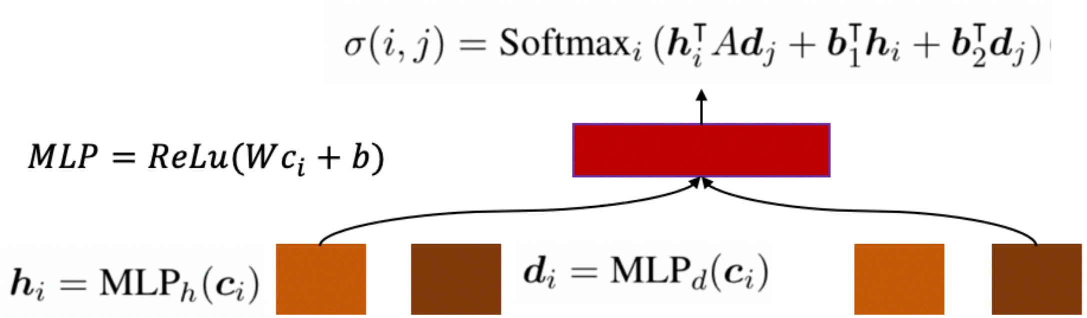
Encoding high-order information

Basic node representations

3.5 Decoding Optimal Tree

- Decoding optimal tree structure
  - Maximum Spanning Tree (MST) Algorithm– Chu-Liu Edmonds Algorithm

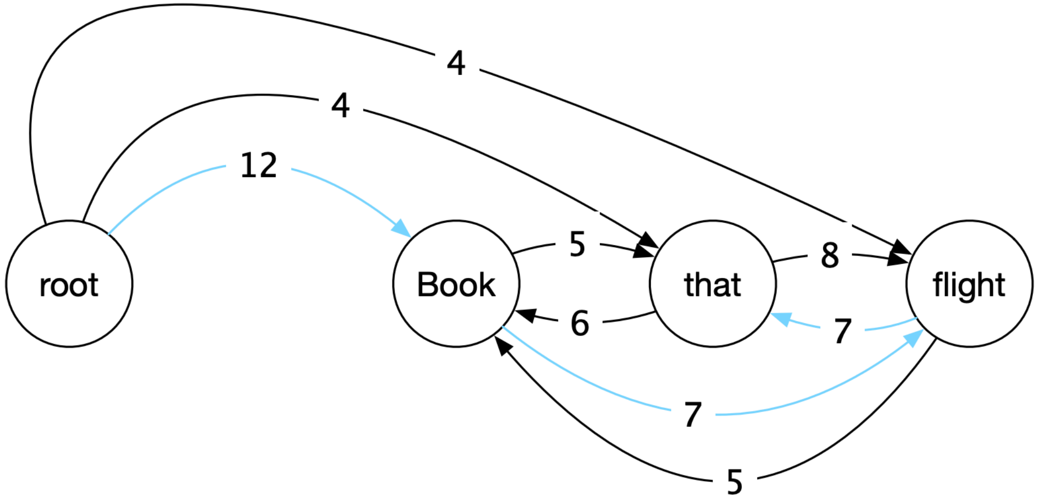
- Label prediction (classification):
  - Similar as computing edge scores.
  - Multi-Layer Perceptron to model P$(r|i,j)$ for edge (i,j)
  - Input: representations for node i and j
  - Output: $r = argmax\ P(r|i,j)$



3.5.1 Maximum Spanning Tree (MST)

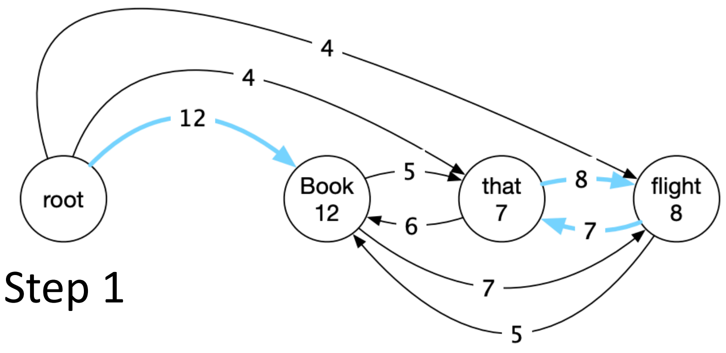
- Spanning Tree: A spanning tree $T(V, E')$ of a directed graph $G = (V, E)$ is a tree over G, which contains all of the nodes in G with minimum number of edges.

- MST: A MST is a spanning tree $T(V, E')$ with highest edge weights.

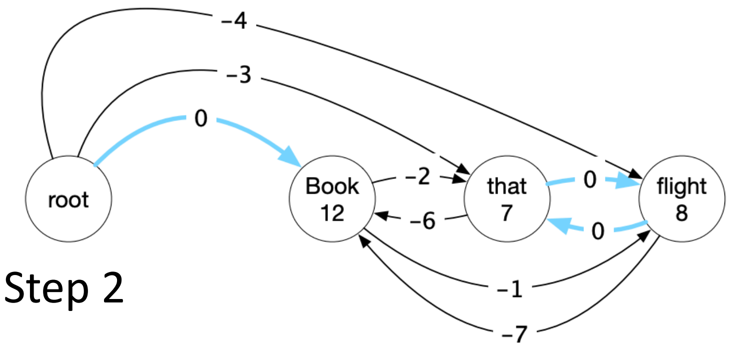


Jurafsky, Daniel, and James H. Martin. "Speech and Language Processing: An introduction to speech recognition, computational linguistics and natural language processing." *Upper Saddle River, NJ: Prentice Hall* (2008).

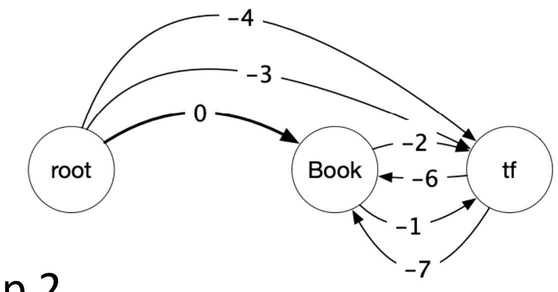
3.5.2 Chu-Liu Edmonds Algorithm



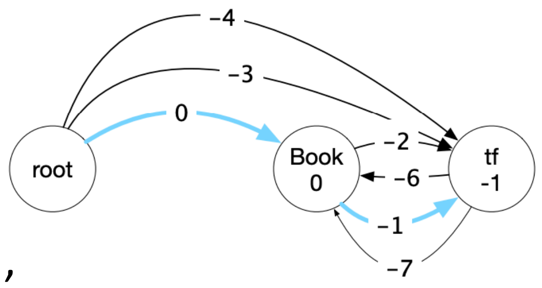
Step 1



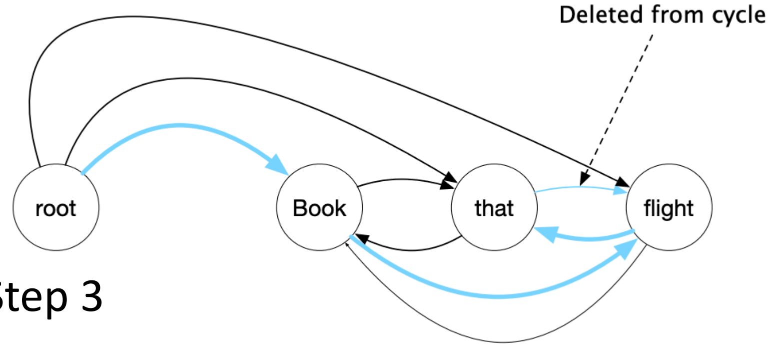
Step 2



Step 2



Step 1’



Step 3

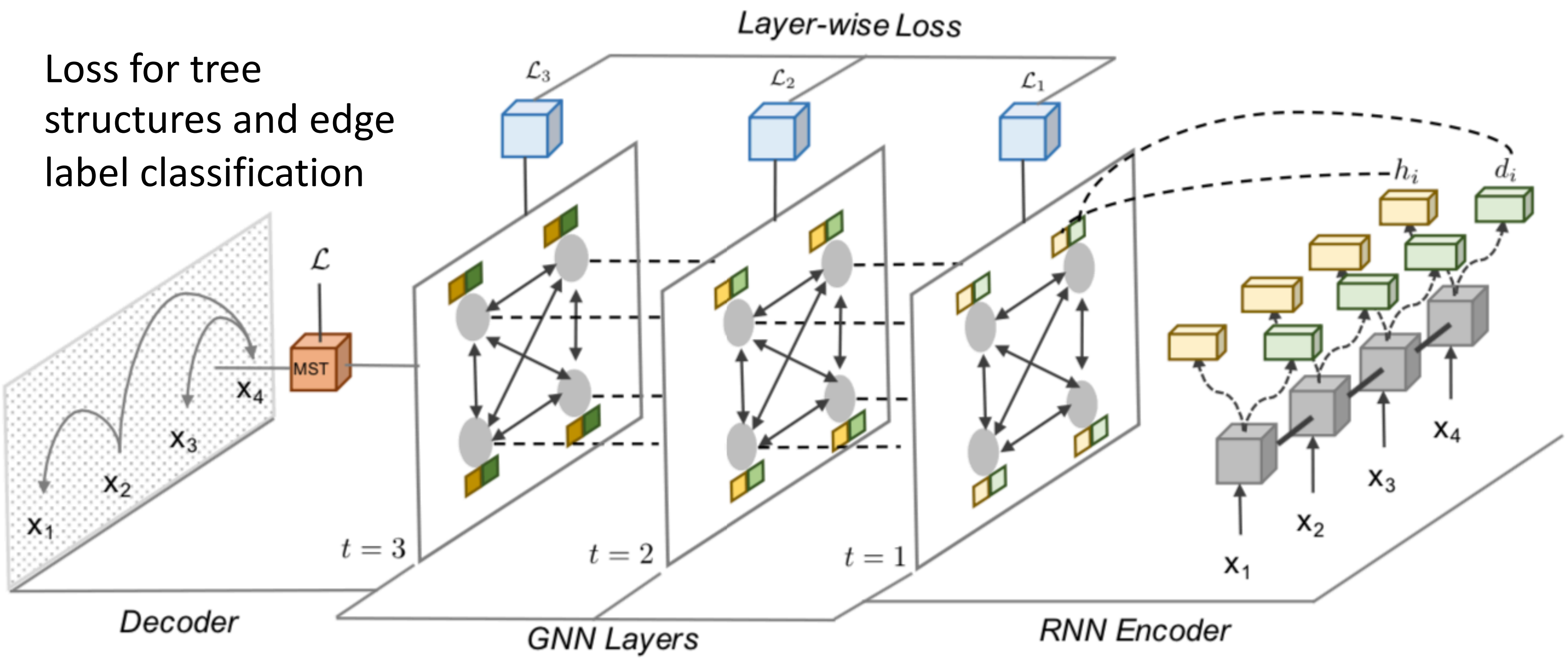
1. Initialization: for each v, choose the incoming edge with highest score.
2. Break cycles: subtract highest scores for all incoming edges, and merge nodes in cycles.
3. Split nodes, and delete edges according to Tree constrain.

Jurafsky, Daniel, and James H. Martin. "Speech and Language Processing: An introduction to speech recognition, computational linguistics and natural language processing." *Upper Saddle River, NJ: Prentice Hall* (2008).

Overview

Loss for predicting edge scores

Loss for tree structures and edge label classification



Infer optimal tree given a complete graph with edge scores.

Encoding high-order information

Basic node representations

3.5.3 Learning Objective

- Loss for the output tree structure
  - Output from the model:
    - tree structure τ
    - probability of edge labels $P(r|i,j)$
    - Probability of edge $P^{\tau}(i|j)$
  - Ground truth: T

n: the number of edges in T

$$\mathcal{L}_0 = -\frac{1}{n} \sum_{(i,j,r)\in T} (\log P^{\tau}(i|j) + \log P(r|i,j))$$

$$L = \lambda_1 \mathcal{L}_0 + \lambda_2 \mathcal{L}'.$$

- Layer-wise loss
  - Edge scores $P^t(i|j)$ for each GCN layer

$$\mathcal{L}' = \sum_{t=1}^{\tau} \mathcal{L}_t = \sum_{t=1}^{\tau} -\frac{1}{n} \sum_{(i,j,r)\in T} \log P^t(i|j)$$

3.6 Experimental Results

- Unlabeled Attachment Score (UAS):
  - The percentage of words that have the corrected **head**.
- Labeled Attachment Score (LAS):
  - The percentage of words that have the corrected **head and label.**
- Unlabeled Complete Match (UCM):
  - The percentage of predicted **trees** that are completely corrected.
- Labeled Complete Match (LCM):
  - The percentage of predicted **trees and labels** are completely corrected.

Jurafsky, Daniel, and James H. Martin. "Speech and Language Processing: An introduction to speech recognition, computational linguistics and natural language processing." *Upper Saddle River, NJ: Prentice Hall* (2008).

3.6.1 Main Results

Parser		Test UAS	Test LAS
(Chen and Manning, 2014)	T	91.8	89.6
(Dyer et al., 2015)	T	93.1	90.9
(Ballesteros et al., 2016)	T	93.56	92.41
(Weiss et al., 2015)	T	94.26	91.42
(Andor et al., 2016)	T	94.61	92.79
(Ma et al., 2018) §	T	95.87	94.19
(Kiperwasser and Goldberg, 2016a) §	G	93.0	90.9
(Kiperwasser and Goldberg, 2016b)	G	93.1	91.0
(Wang and Chang, 2016)	G	94.08	91.82
(Cheng et al., 2016)	G	94.10	91.49
(Kuncoro et al., 2016)	G	94.26	92.06
(Zheng, 2017) §	G	95.53	93.94
(Dozat and Manning, 2017)	G	95.74	94.08
Baseline	G	95.68	93.96
Our Model §	G	**95.97**	**94.31**

Table 1: Results on the English PTB dataset. The § indicates parsers using high-order features. "T" represents transition-based parser, and "G" represents a graph-based parser.

3.6.2 Different Layers and Updating Methods

GNN Layer	GNN Model	Dev UAS	Dev LAS	Test UAS	Test LAS
$l = 0$	Baseline	95.58	93.74	95.68	93.96
$l = 1$	$d \triangleright h$	95.75	93.84	95.83	94.15
	$h \triangleright h$	**95.78**	93.80	**95.91**	94.12
	$hd \triangleright h$	95.77	**93.87**	95.88	**94.23**
$l = 2$	$d \triangleright h$	**95.80**	93.85	95.88	94.17
	$h \triangleright h$	95.77	93.83	95.85	94.13
	$hd \triangleright h$	95.79	**93.90**	**95.92**	**94.24**
$l = 3$	$d \triangleright h$	95.74	93.78	95.87	94.14
	$h \triangleright h$	**95.75**	93.80	95.90	94.15
	$hd \triangleright h$	95.71	**93.82**	**95.93**	**94.22**

Table 2: Impact of l and different high-order information integration methods on PTB dataset. "$d \triangleright h$" corresponds to the Equation 7, "$h \triangleright h$" corresponds to the Equation 6, "$hd \triangleright h$" corresponds to the Equation 8.

GNN Layer	GNN Model	Dev UAS	Dev LAS	Test UAS	Test LAS
$l = 2$	Synch	95.79	93.90	95.92	94.24
	H-first	**95.88**	**93.94**	**95.97**	**94.31**
	D-first	95.78	93.91	95.95	94.27

Table 3: Impact of different GNN update methods on PTB dataset. "Synch" is our default synchronized setting (Equation 8). "H-first" is an asynchronous update method that first updates head word representation (Equation 9). Similarly, the "D-first" model first updates dependent word representation.

3.6.2 Different Layers and Updating Methods

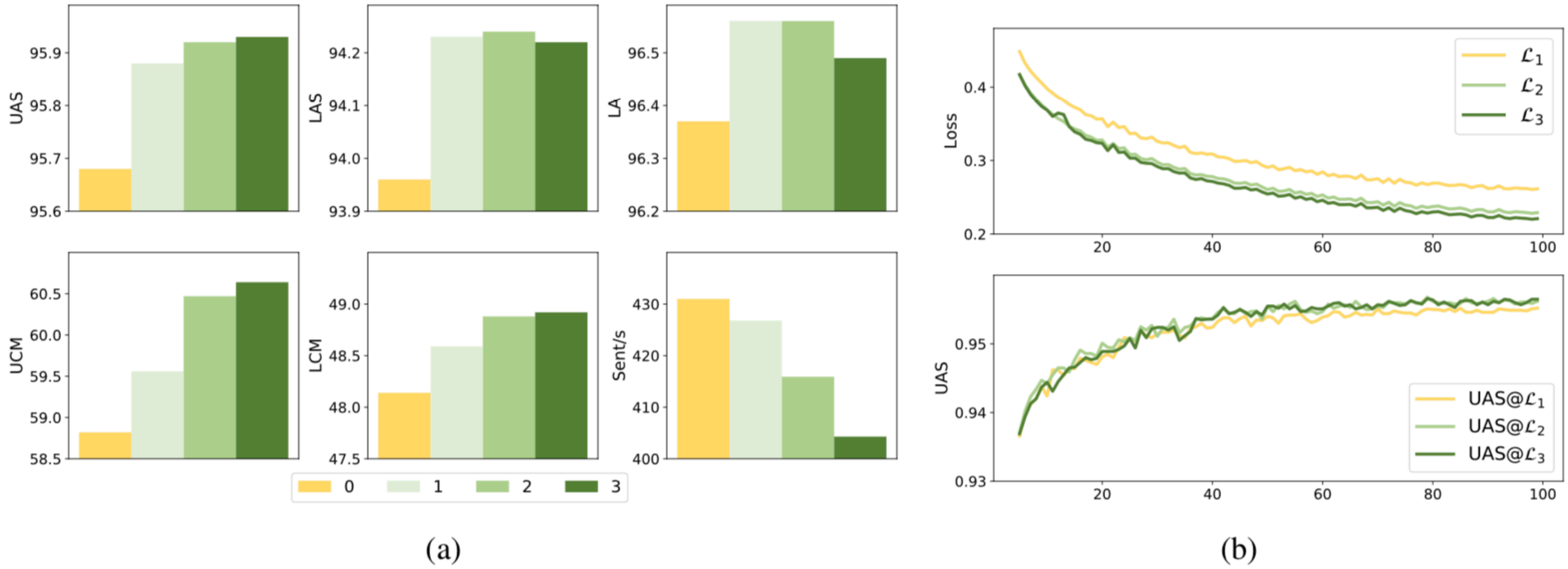


Figure 3: (a) Parsing performance and speed of different layers of our $hd \triangleright h$ model on the test set. (b) Layer-wise training loss and development set's UAS of our 3-layer $hd \triangleright h$ model.

3.6.3 Different Scoring Functions

GNN Layer	GNN Model	Dev UAS	Dev LAS	Test UAS	Test LAS
$l = 2$	All=1	95.71	93.73	95.76	94.07
	Hard-1	95.69	93.70	95.80	94.13
	Hard-2	95.73	93.78	**95.90**	**94.20**
	Hard-3	**95.81**	**93.88**	95.88	**94.20**
$l = 2$	Soft	**95.88**	**93.94**	**95.97**	**94.31**

Table 4: Impact of different kinds of graph weights on PTB dataset. "All=1" means setting all weights to 1 (Equation 12), "Hard-k" means renormalization at the top-k weights of each node (Equation 11), "Soft" is our default model setting (Equation 8).

3.6.4 Error Analysis

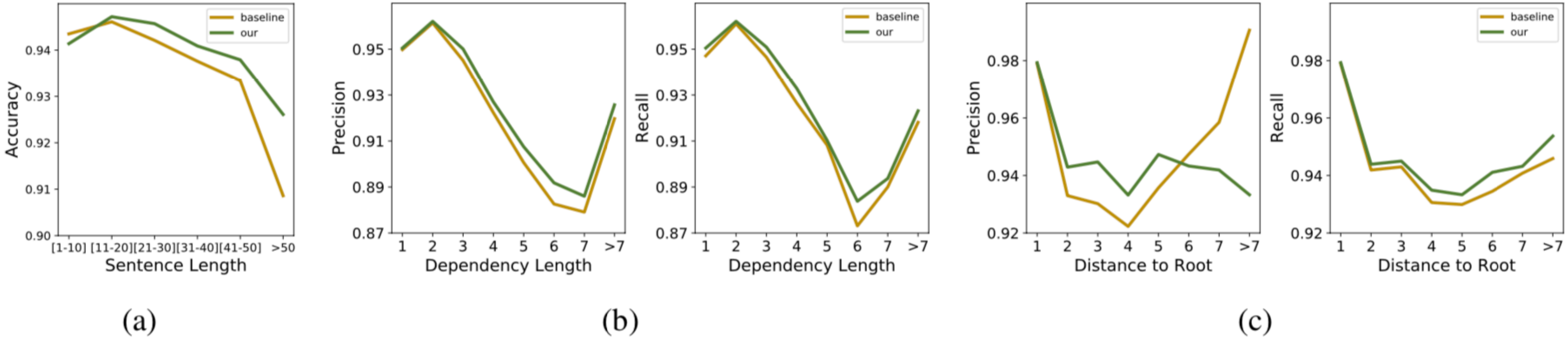


Figure 4: Parsing performance of baseline and our best parser relative to length and graph factors.

References

1. Jurafsky, Daniel, and James H. Martin. "Speech and Language Processing: An introduction to speech recognition, computational linguistics and natural language processing." *Upper Saddle River, NJ: Prentice Hall* (2008).
2. Marcus, Mitchell, et al. "The Penn Treebank: annotating predicate argument structure." *Proceedings of the workshop on Human Language Technology*. Association for Computational Linguistics, 1994.
3. Ji, Tao, Yuanbin Wu, and Man Lan. "Graph-based dependency parsing with graph neural networks." *Proceedings of ACL*. 2019.