A I L A B S E M I N A R

Graph Convolution over Pruned Dependency Trees Improves Relation Extraction

석사 2기 조충현

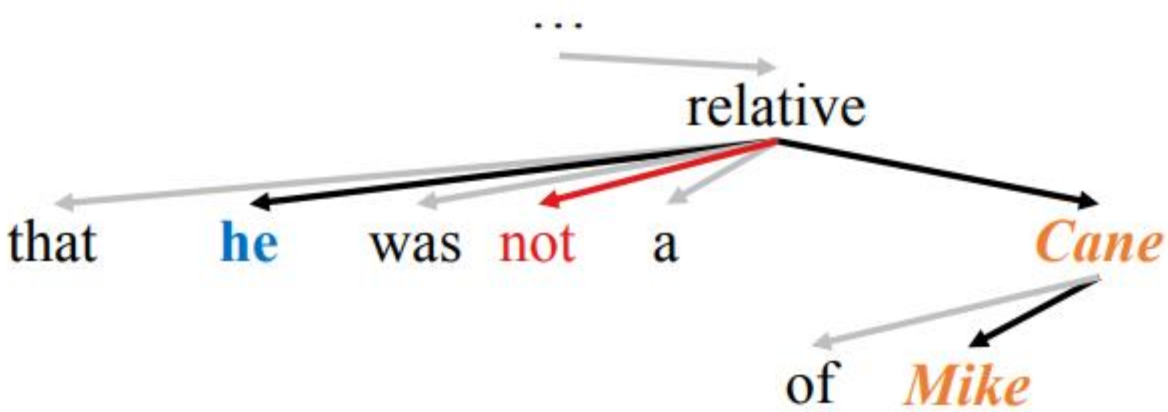
CONTENTS

- **Dependency tree**는 **relation extraction** 모델에 **word**간 사이가 큰 범위의 관계를 찾는데 도움을 준다

- 하지만 기존 **Dependency-based model**들은 **dependency** 트리를 너무 공격적으로 잘라내서 중요한 정보를 무시하거나 다른 트리 구조와 병렬화하기 어려워서 계산이 복잡한 단점이 있다

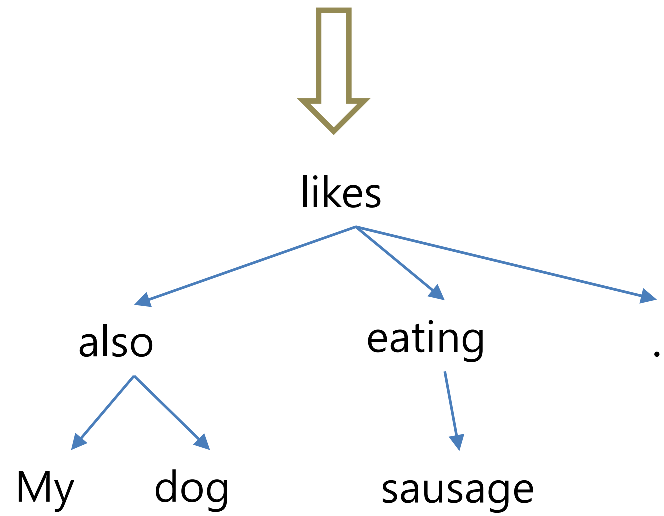
- 그래서 이 논문에서는 **relation extraction**에 맞춘 확장된 **Graph convolution network** 모델을 제안
  - ✓ 불필요한 정보는 최대한 제거하면서 필요한 정보를 통합
  - ✓ 두 엔티티간 최단 경로에 있는 단어들을 유지하는 새로운 **pruning** 전략을 제안

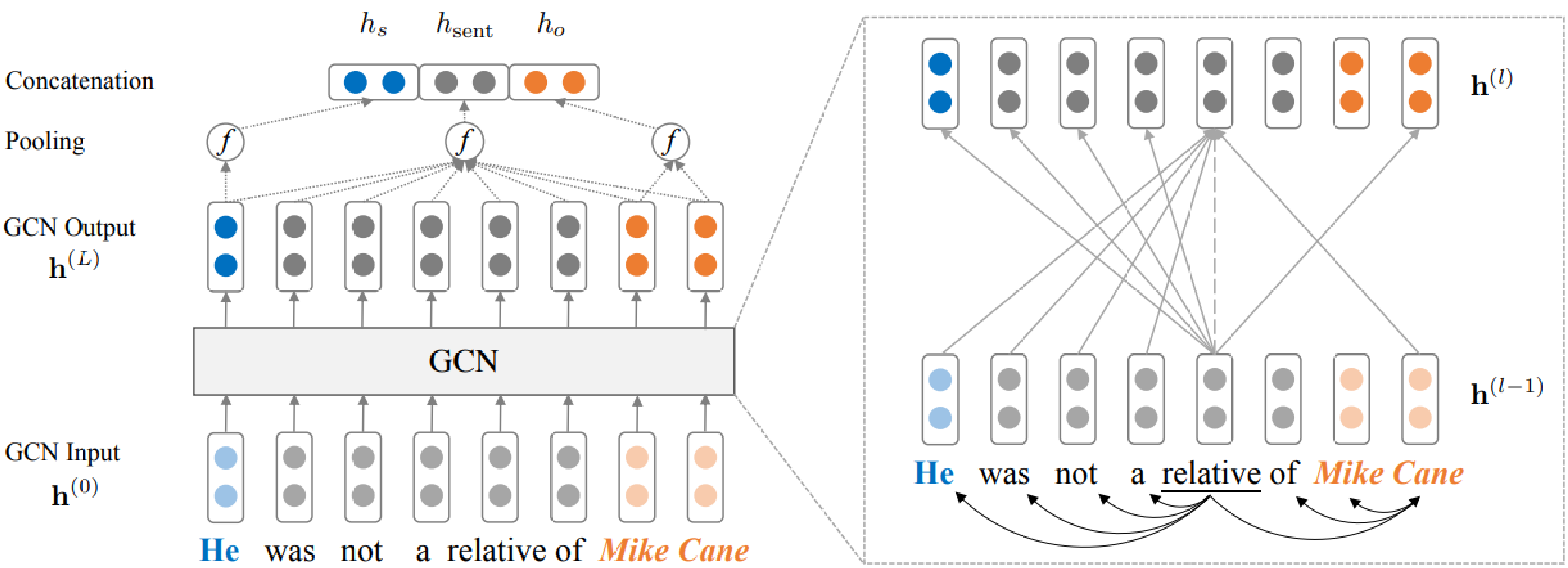
I had an e-mail exchange with Benjamin Cane of Popular Mechanics which showed that **he** was not a relative of *Mike Cane*.



Prediction from dependency path: *per:other_family*
Gold label: *no_relation*

My dog also likes eating sausage.




h_s h_{sent} h_o
Concatenation
Pooling
f
GCN Output
$\mathbf{h}^{(L)}$
GCN
GCN Input
$\mathbf{h}^{(0)}$
He was not a relative of Mike Cane
$\mathbf{h}^{(l)}$
$\mathbf{h}^{(l-1)}$
He was not a relative of Mike Cane

$$h_i^{(l)} = \sigma\left(\sum_{j=1}^{n} A_{ij} W^{(l)} h_j^{(l-1)} + b^{(l)}\right)$$

문제점

- 각 노드별로 **degree**가 다르기 때문에 높은 **degree**로 치우치는 문제 발생
- 또한 인접행렬이기 때문에 자기자신의 정보를 전달하지 않는 문제 발생

$$\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$$

인접행렬에 단위행렬을 더해서 **self-loop** 추가

$$d_i = \sum_{j=1}^{n} \tilde{A}_{ij}$$

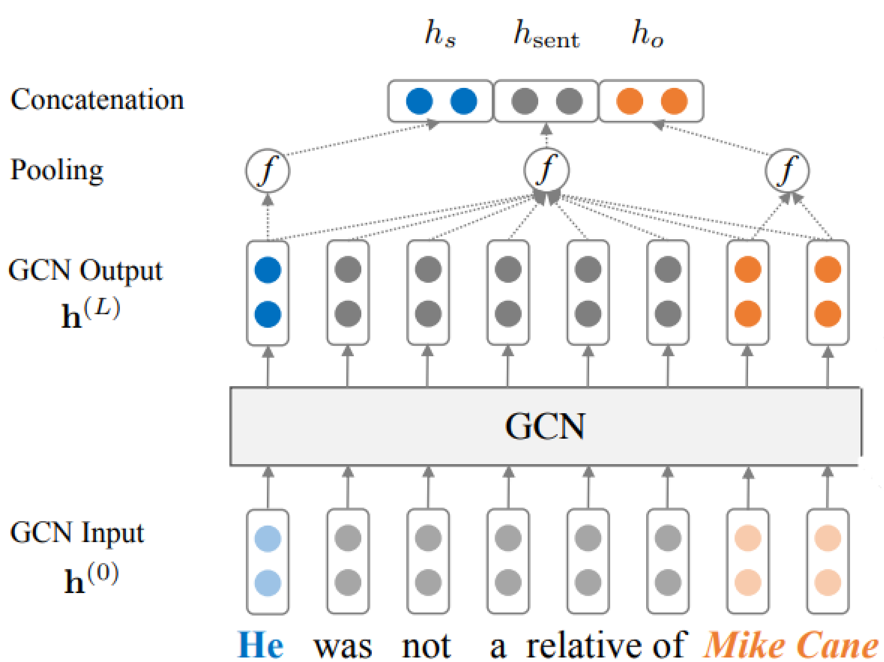
Normalizing을 통하여 해결

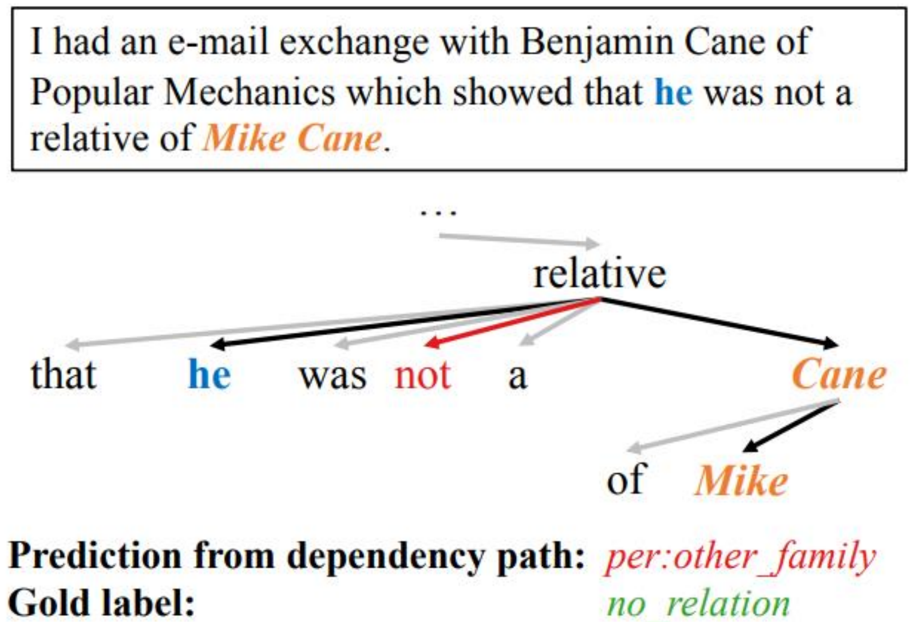
$$h_i^{(l)} = \sigma\left(\sum_{j=1}^{n} \tilde{A}_{ij} W^{(l)} h_j^{(l-1)} / d_i + b^{(l)}\right)$$

$$h_{\text{sent}} = f\left(\mathbf{h}^{(L)}\right) = f\left(\text{GCN}(\mathbf{h}^{(0)})\right)$$

$$h_s = f\left(\mathbf{h}^{(L)}_{s_1:s_2}\right)$$

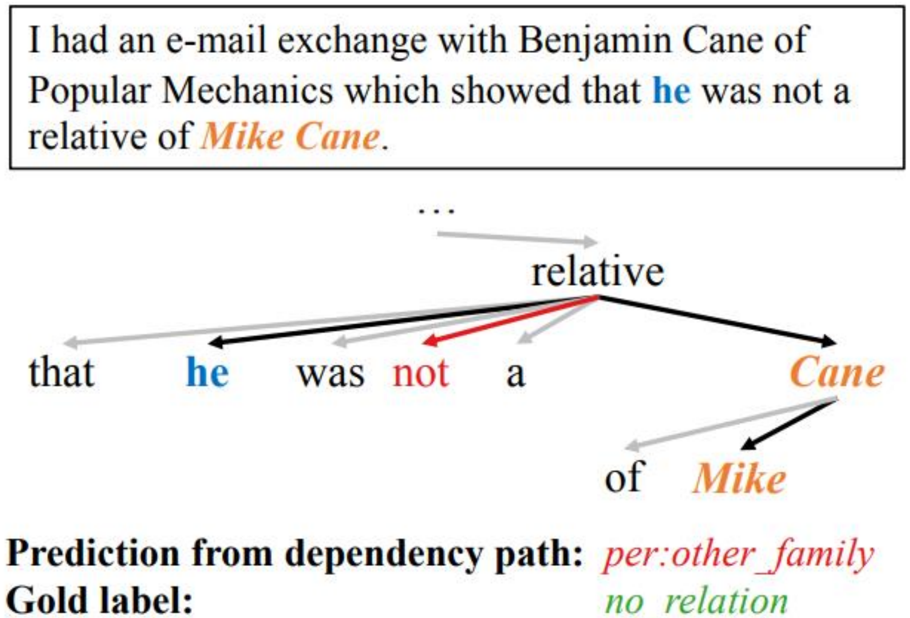
$$h_{\text{final}} = \text{FFNN}\left([h_{\text{sent}}; h_s; h_o]\right)$$





두 엔티티간의 공통조상까지의 경로에 있는 노드들이 **relation**을 나타내는데 적절한 정보를 가지고 있고, 이전 연구에서 **LCA** 범위 외에 있는 토큰들을 제거하는 것이 불필요한 정보를 제거하는 것이라고 검증되었다

따라서**!** 이 **pruning** 전략을 **GCN**에 적용 시키겠다**!**



그러나 너무나 공격적으로 **pruning**을 하면 중요한 정보까지 잃게 되는 문제가 있다!

그래서! **LCA**경로에서 **K**거리만큼 떨어진 노드까지만 포함시키는 새로운 **pruning**전략을 제안했다

System	P	R	F_1
LR† (Zhang+2017)	**<u>73.5</u>**	49.9	59.4
SDP-LSTM† (Xu+2015b)	66.3	52.7	58.7
Tree-LSTM‡ (Tai+2015)	66.0	59.2	62.4
PA-LSTM† (Zhang+2017)	65.7	<u>64.5</u>	65.1
GCN	69.8	59.0	64.0
C-GCN	69.9	63.3	<u>66.4</u>*
GCN + PA-LSTM	71.7	63.0	67.1*
C-GCN + PA-LSTM	71.3	**65.4**	**68.2***

Table 1: Results on TACRED. Underscore marks highest number among single models; bold marks highest among all. † marks results reported in (Zhang et al., 2017); ‡ marks results produced with our implementation. ∗ marks statistically significant improvements over PA-LSTM with $p < .01$ under a bootstrap test.

System	*with-m*	*mask-m*
SVM† (Rink+2010)	82.2	–
SDP-LSTM† (Xu+2015b)	83.7	–
SPTree† (Miwa+2016)	84.4	–
PA-LSTM‡ (Zhang+2017)	82.7	75.3
Our Model (C-GCN)	**84.8***	**76.5***

Table 2: F_1 scores on SemEval. † marks results reported in the original papers; ‡ marks results produced by using the open implementation. The last two columns show results from *with-mention* evaluation and *mask-mention* evaluation, respectively. ∗ marks statistically significant improvements over PA-LSTM with $p < .05$ under a bootstrap test.

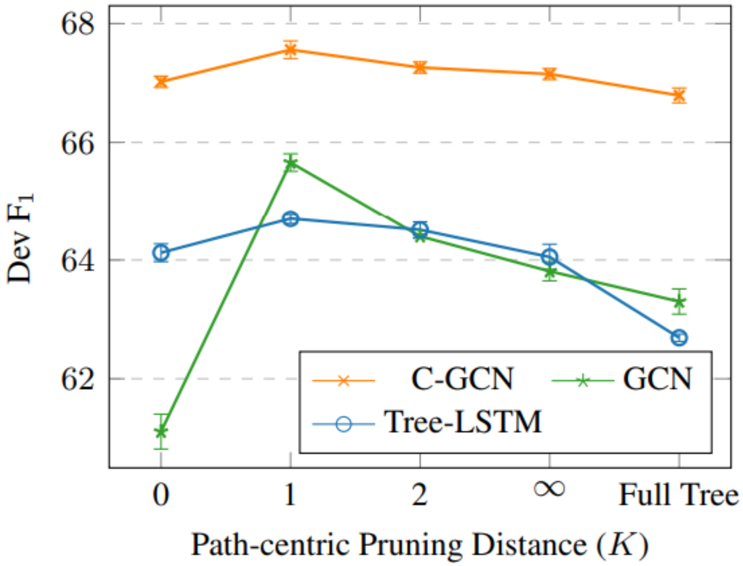


Figure 3: Performance of dependency-based models under different pruning strategies. For each model we show the F_1 score on the TACRED dev set averaged over 5 runs, and error bars indicate standard deviation of the mean estimate. $K = \infty$ is equivalent to using the subtree rooted at the LCA.

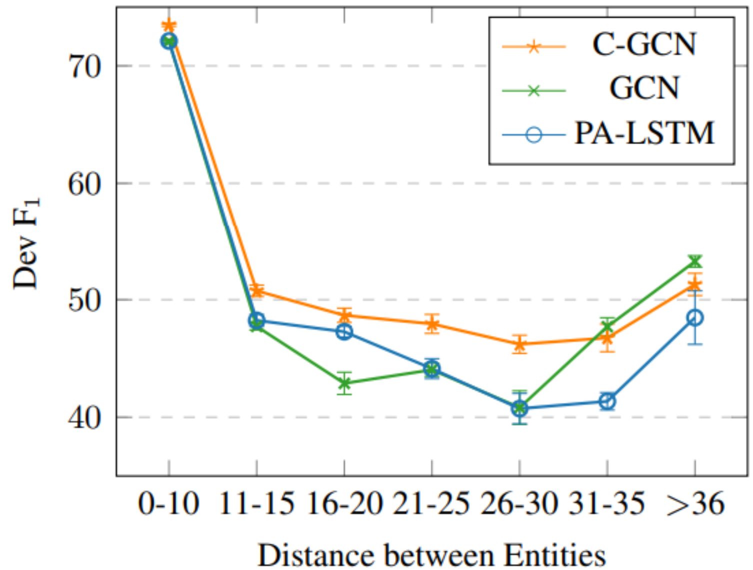


Figure 4: Dev set performance with regard to distance between the entities in the sentence for C-GCN, GCN and PA-LSTM. Error bars indicate standard deviation of the mean estimate over 5 runs.

Model	Dev F_1
Best C-GCN	67.4
– h_s, h_o, and Feedforward (FF)	66.4
– LSTM Layer	65.5
– Dependency tree structure	64.2
– FF, LSTM, and Tree	57.1
– FF, LSTM, Tree, and Pruning	47.4

Table 3: An ablation study of the best C-GCN model. Scores are median of 5 models.

THANK YOU

석사 2기 조충현