KIPS Trans. Softw. and Data Eng.
Vol.10, No.6 pp.235~242
ISSN: 2287-5905 (Print), ISSN: 2734-0503 (Online)

https://doi.org/10.3745/KTSDE.2021.10.6.235

Korean Dependency Parsing Using Stack-Pointer Networks and Subtree Information

Yong-Seok Choi† · Kong Joo Lee††

ABSTRACT

In this work, we develop a Korean dependency parser based on a stack-pointer network that consists of a pointer network and an internal stack. The parser has an encoder and decoder and builds a dependency tree for an input sentence in a depth-first manner. The encoder of the parser encodes an input sentence, and the decoder selects a child for the word at the top of the stack at each step. Since the parser has the internal stack where a search path is stored, the parser can utilize information of previously derived subtrees when selecting a child node. Previous studies used only a grandparent and the most recently visited sibling without considering a subtree structure. In this paper, we introduce graph attention networks that can represent a previously derived subtree. Then we modify our parser based on the stack-pointer network to utilize subtree information produced by the graph attention networks. After training the dependency parser using Sejong and Everyone's corpus, we evaluate the parser's performance. Experimental results show that the proposed parser achieves better performance than the previous approaches at sentence-level accuracies when adopting 2-depth graph attention networks.

Keywords : Stack-Pointer Networks, Korean Dependency Parser, Graph Attention Networks, Subtree, Pre-trained Word Representation Model, Everyone's Corpus

스택-포인터 네트워크와 부분 트리 정보를 이용한 한국어 의존 구문 분석

최 용 석† · 이 공 주††

요 약

본 연구에서는 포인터 네트워크 모델을 의존 구문 분석에 맞게 확장한 스택-포인터 네트워크 모델을 이용하여 한국어 의존 구문 분석기를 구현한다. 스택-포인터 네트워크 모델 기반 의존 구문 분석기는 인코더-디코더로 구성되어 있으며 다른 의존 구문 분석기와 달리 내부 스택을 갖고 있어 루트부터 시작하는 하향식 구문 분석이 가능하다. 디코더의 각 단계에서는 의존소를 찾기 위해 부모 노드뿐만 아니라 이미 파생된 트리 구조에서 조부모와 형제 노드를 참조할 수 있다. 기존 연구에서는 단순하게 해당 노드들의 합을 계산하여 입력으로 사용하였고, 형제 노드의 경우에는 가장 최근에 방문했던 것만을 사용할 수 있었다. 본 연구에서는 그래프 어텐션 네트워크를 도입하여 이미 파생된 부분 트리를 표현하고 이를 스택-포인터 네트워크의 입력으로 사용하도록 구문 분석기를 수정한다. 세종 코퍼스와 모두의 코퍼스를 대상을 실험한 결과 레이어 2의 그래프 어텐션 네트워크를 이용하여 부분 트리를 표현했을 때 특히 문장 단위의 구문 분석 정확도에서 많은 성능 향상을 확인할 수 있었다.

키워드 : 스택-포인터 네트워크, 한국어 의존 구문 분석, 그래프 어텐션 네트워크, 부분 트리, 사전 훈련된 단어 표현, 모두의 코퍼스

1. 서 론

의존 구문 분석은 문장에서 각 단어에 대한 의존소(modifier)와 지배소(head) 사이의 의존 여부를 파악하는 작업이다. 한국어는 교착어이면서 어순 배열의 자유도가 높고 문장 성분의 생략이 빈번한 특성 때문에 의존 구문 분석이 주로 많이 사용되었다[1-8]. 한국어 의존 구문 트리의 규칙은 영어권 연구에서처럼 단일 지배소 원칙과 투사성 원칙을 적용하고 한국어의 고유 특성을 반영하기 위해 지배소 후위 원칙을 적용하였다[9]. 의존 구문 분석의 주된 방법은 그래프와 전이 기반으로 나뉜다. 그래프 기반의 의존 구문 분석은 모든 단어 쌍에 대해 의존 가능성을 점수로 표현하고 가장 높은 점수를 가지는 의존 트리를 찾는다. 전이 기반의 의존 구문 분석은

※ 이 논문은 2019년 대한민국 교육부와 한국연구재단의 지원을 받아 수행된 연구임(NRF-2019S1A5A2A03041296).
† 준 회 원 : 충남대학교 전자전파정보통신공학과 박사과정
†† 종신회원 : 충남대학교 전파정보통신공학과 교수
Manuscript Received : January 5, 2021
Accepted : February 4, 2021
* Corresponding Author : Kong Joo Lee(kjoolee@cnu.ac.kr)

두 단어의 의존 여부(전이 액션)를 결정하는 방법과 어텐션을 계산해서 각 단어의 지배소 또는 의존소를 찾는다. 이 방법은 탐욕적(greedy)인 방식으로 의존 트리를 구성해 나간다.

그래프 기반 방식의 대표적인 모델은 Biaffine 어텐션 모델[10]이다. [10]의 모델은 양방향 재귀 신경망을 통해서 문장을 인코딩한다. 인코딩된 문장으로부터 각 단어를 지배소와 의존소로 표상화시킨다. 표상화된 지배소와 의존소를 입력으로 Biaffine 어텐션을 이용해 모든 단어 쌍에 대해 점수를 계산한다. 계산된 점수는 의존 여부를 결정할 때 사용된다.

전이 기반 방식의 대표적인 모델은 두 단어의 의존 여부를 전이 액션으로 분류하는 모델[11]과 어텐션을 이용한 포인터 네트워크 모델[12]이다. [11]의 모델은 스택과 버퍼 그리고 전이 액션을 각각 재귀 신경망을 통해 표상화시키고 이를 통해 전이 액션을 결정한다. 포인터 네트워크[13]의 의존 구문 분석 모델은 전이 액션을 결정하는 것이 아닌 어텐션을 계산해서 해당 단어의 지배소 또는 의존소를 가리킨다. [12]의 모델은 포인터 네트워크[13] 기반의 내부 스택을 포함한 스택-포인터 네트워크이다. 스택-포인터 네트워크는 포인터 네트워크처럼 인코더와 디코더로 이루어져 있다. 인코더는 입력 문장의 단어를 압축하여 인코딩한다. 디코더는 인코딩된 단어들을 입력과 인코더 출력된 표현들 사이의 어텐션을 계산해 의존소를 찾는다. 스택-포인터 네트워크는 각 단어의 의존소를 찾는 하향식 방식이며 부모 노드의 순서를 유지하기 위한 내부 스택을 갖추고 있다. [11]의 모델은 오직 두 단어의 관계(지배소-의존소, 1st order)만 보고 의존 관계를 결정하는 반면에 [12]에서 제안한 모델은 형제, 조부모 정보(2nd order)를 디코더의 입력에 추가해 의존소를 찾는다. 그렇지만 [12]의 모델은 내부 스택 특성으로 가장 최근에 찾은 형제 노드만을 입력으로 사용할 수 있다.

[14]의 연구에서는 biaffine 어텐션 모델을 기반으로 형제, 조부모, 손자 노드의 정보를 이용하는 그래프 신경 네트워크(Graph Neural Networks)를 제안하여 좋은 성능을 보였다. 그러나 [14]의 연구는 그래프 기반 방식으로 모든 쌍에 대해 계산하기 때문에 명시적으로 지금까지 만들어진 부분 의존 트리가 제공될 수 없다.

본 연구에서는 [12]의 연구를 기반으로 이미 만들어진 부분 트리 정보를 반영할 수 있는 모델을 제안한다. 트리 정보는 각 단계마다 만들어진 의존 트리를 이용한다. 트리 정보를 반영하기 위한 모델은 그래프 어텐션 네트워크(Graph Attention Networks)[15]를 활용한다. 그래프 어텐션 네트워크는 노드의 연결(Adjacency) 정보를 담고 있는 매트릭스를 이용해 해당 노드와 연결된 노드들 사이의 어텐션을 계산해서 해당 노드를 표현하는 모델이다. 본 연구에서는 그래프 어텐션 네트워크가 연결된 노드들에 대해서만 계산하기 때문에 여러 개의 레이어를 사용해 각 노드의 표현을 N차 연결된 노드까지 반영할 수 있도록 한다. 이를 통해 각 단계마다 만들어진 트리 정보가 부분적으로 입력에 반영될 수 있도록 한다. 추가적으로 인코더에서 문맥 표현을 풍부하게 해주기 위해 사전 훈련된 언어 모델인 BERT를 결합해서 의존 구문 분석 모델을 제안한다.

2. 관련 연구

최근 한국어 의존 구문 분석 연구에서는 기존 모델에 사전 훈련된 언어 모델을 결합한 연구가 활발히 이루어지고 있다[6-8].

대표적인 사전 훈련된 언어 모델로는 BERT[16]와 RoBERTa[17] 등이 있다. 두 모델은 트랜스포머의 인코더를 기반으로 언어적 이해를 할 수 있도록 사전 훈련시킨 모델이다. BERT 모델[16]은 주석이 없는 대량의 코퍼스로부터 사전 훈련을 시킨다. 사전 훈련하는 방법은 두 가지로 나뉜다. 첫째, 문장에서 임의의 토큰을 마스킹 토큰으로 교체해서 입력으로 넣어주고 이를 원래 토큰으로 예측하는 방법(MLM)이다. 둘째, 두 개의 세그먼트가 서로 연속이거나 연속되지 않은 쌍을 입력으로 넣어주었을 때, 입력이 연속된 세그먼트인지 아닌지를 분류하는 방법(NSP)이다.

RoBERTa 모델[17]은 BERT 모델의 학습 방법을 약간 바꿔서 모델이 더 강건해질 수 있도록 하였다. 강건한 모델을 위하여 사전 훈련에 사용되는 코퍼스를 확장하고 배치 크기를 늘렸다. 또한 기존의 BERT 모델에서 사용한 MLM 학습 방법을 수정하여 매번 학습을 진행할 때마다 마스킹 토큰의 위치가 바뀔 수 있도록 했다. 그리고 NSP 방식은 학습에서 제거하였다.

[6-8] 연구 모두 사전 훈련된 언어 모델을 사용하는 그래프 기반의 구문 분석기이다. [6,8] 연구는 Biaffine 어텐션 모델을 기반으로 BERT[16] 모델을 결합했고 [7] 연구는 RoBERTa[17] 모델을 결합했다.

[6] 연구는 형태소로 분석된 문장을 워드피스[18]로 토큰화하고 이를 BERT 모델의 입력으로 사용한다. BERT 모델로부터 워드피스들의 표상을 얻고 이를 기반으로 다시 양방향 LSTM의 입력에 사용한다. 이 때, 양방향 LSTM의 입력에는 어절 범위의 특성을 표현하는 임베딩과 형태소 범위의 특성을 표현하는 임베딩을 같이 넣어주었다. 실험 결과로는 94.06(UAS), 92.00(LAS)의 성능을 보였다. [7]의 연구는 RoBERT 모델로부터 워드피스들의 표상을 얻고 어절의 표상을 얻기 위해 어절 내의 형태소와 워드피스의 마지막 토큰을 결합하였다. 실험 결과로는 94.32(UAS), 92.40(LAS)의 성능을 보여주었다.

[8] 연구는 [14] 모델을 기반으로 해서 BERT 모델을 결합했다. 형제, 조부모, 손자 정보를 표현할 수 있는 수식을 정의하고 이에 대해 모든 경우의 쌍에 대한 점수를 계산했다. 계산된 노드 쌍 중에서 가장 높은 점수를 트리로 구성했다. 실험 결과로는 94.44(UAS), 92.55(LAS)의 성능을 보였다. [8]의 연구처럼 트리 정보를 반영하고자하는 시도는 있었다. 하지만 [8]의 연구에서는 트리 정보를 명시적으로 반영하지는 않았다.

본 연구에서는 스택-포인터 네트워크를 기반으로 부분 트

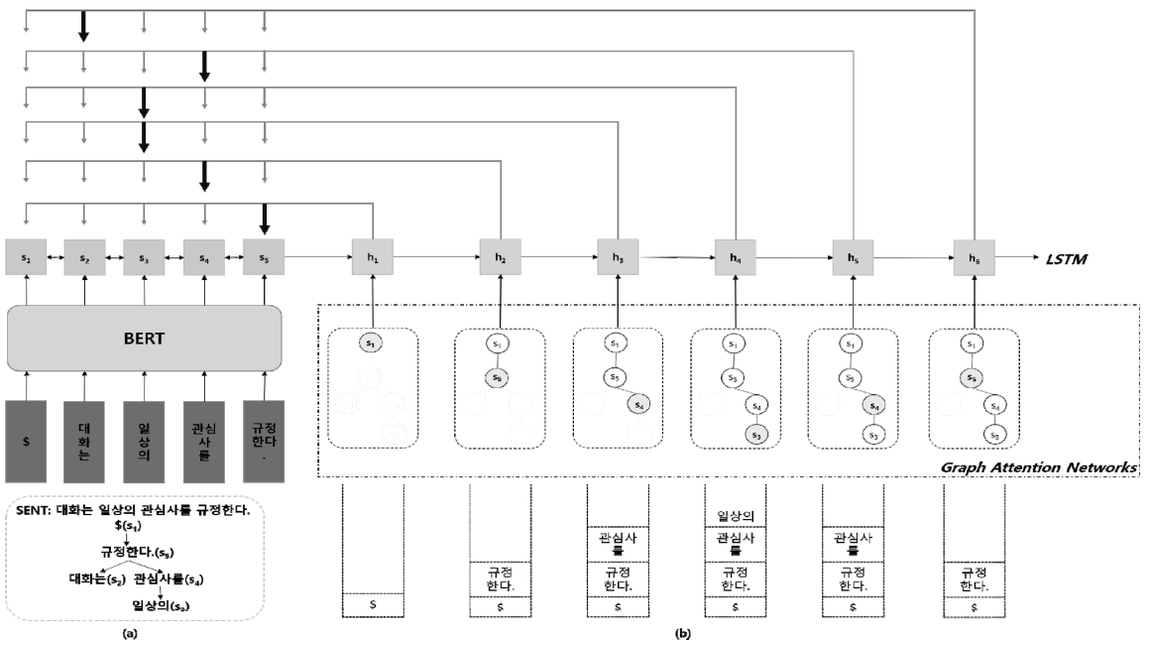

Fig. 1. Dependency Parser using Stack-pointer Networks and Graph Attention Networks

리 정보를 반영한 의존 구문 분석 모델을 제안한다. 또한 [6-8] 연구와 같이 사전 훈련된 언어 모델을 결합해 문장 내의 단어들이 풍부한 정보를 반영할 수 있도록 한다.

3. 트리 정보를 반영한 스택-포인터 네트워크 모델

본 연구에서는 [12]에서 제안한 스택-포인터 네트워크(Stack-Pointer Networks) 모델을 기반으로 구문 분석기를 구현한다. 스택-포인터 네트워크는 인코더와 디코더로 구성된다. 인코더는 문장을 구성하고 있는 단어를 압축하여 인코딩하고 디코더는 각 단계마다 순서 정보를 포함한 표현을 생성한다. 디코더에서 생성한 표현과 인코더에서 생성한 표현 사이의 어텐션을 계산해 가장 높은 값을 갖는 인코더의 단어를 의존소 관계로 정의한다. 스택-포인터 네트워크 모델은 디코더의 입력을 관리하는 내부 스택이 포함되어 있어 다른 의존 구문 파서들과 다르게 내부 스택을 통해 해당 단어의 의존소를 찾는 하향식 접근 방식 모델을 채택하였다. 그리고 의존소를 찾기 위해 부모 노드뿐만 아니라 조부모와 형제 노드를 입력에 같이 포함하였다. 그렇지만 단순히 노드들의 합을 계산해 입력으로 사용하였고 형제 노드의 경우에는 가장 최근에 방문했던 것만을 사용할 수 있다는 한계가 있다. 본 연구에서는 각 단계마다 만들어진 트리 정보를 반영하기 위해 스택-포인터 네트워크의 디코더 입력으로 그래프 어텐션 네트워크를 사용한다(Fig. 1).

3.1 스택-포인터 네트워크 기반의 한국어 의존 구문 분석

스택-포인터 네트워크[12] 모델은 포인터 네트워크[13]를 기반으로 인코더와 디코더로 구성되어 있다. 추가로 디코더의 입력을 관리하는 내부 스택이 포함되어 있다.

인코더는 문장을 구성하고 있는 단어를 입력으로 주어져서 입력열의 위치에 해당하는 단어들의 표현들을 생성한다.

본 연구에서는 단어 사이의 문맥을 고려해 주기 위해 BERT[16] 모델로부터 단어 임베딩을 생성한다(z_t). BERT의 입력 단위는 워드피스(WordPiece)[18]를 사용하기 때문에 BERT에서 생성한 단어 임베딩(X)을 한국어 의존 구문 분석에 맞게 어절(W) 단위로 바꾼다. 어절 임베딩($\hat{z}_t$)은 어절 레벨에서의 문맥을 고려해 주기 위해 LSTM의 입력으로 주어져서 최종 어절 임베딩(s_t)으로 표현한다[Equation (1)].

$$
\begin{gathered}
X=\{x_1, x_2, .., x_N\},\ W=\{w_1, w_2, ..., w_M\} \\
z_t = BERT(X) \\
\hat{z_i} = \sum_{z_t \in w_i} z_t \\
\vec{s}_t = LSTM(\hat{z}_t, \vec{s}_{t-1}),\ \overleftarrow{s}_t = LSTM(\hat{z}_t, \overleftarrow{s}_{t-1}) \\
s_t = [\vec{s}_t \parallel \overleftarrow{s}_t]
\end{gathered} \tag{1}
$$

디코더는 순서 정보를 포함한 표현을 생성한다. 디코더의 입력은 각 단계마다 내부 스택에서 가장 마지막에 들어간 어절로 인코더에서 생성한 해당 어절의 임베딩을 사용한다.

고차원 정보를 반영하기 위해 [12] 연구에서는 Fig. 2와 같이 의존소와 의존소의 형제 관계, 조부모 관계에 해당하는 어절의 임베딩을 사용한다. [12]의 디코더 입력은 각 단계마다 지배소와 조부모, 형제에 해당하는 어절 임베딩을 더해서 사용한다[Equation (2)].

$$
\beta_t = s_h + s_g + s_s \tag{2}
$$

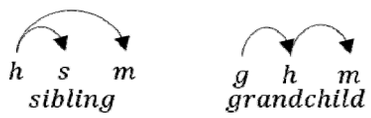

Fig. 2. Higher-order Inputs of Decoder[12]

내부 스택은 각 단계마다 찾은 의존소에 해당하는 어절을 스택에 넣어준다. 다시 이 의존소는 다음 단계의 부모 노드로서 디코더의 입력이 된다. 만약 입력과 출력의 단어가 같은 경우에는 스택에 들어있는 어절을 꺼내 주고 다음 단계의 형제 정보로 사용한다. [12] 모델에서는 내부 스택 특성 때문에 가장 이전 단계에 존재하는 형제 노드만을 사용할 수 있다. 이러한 문제는 지금까지 찾은 형제 정보를 전부 반영할 수 없으며 각 단계마다 만들어진 트리에 대한 정보를 모두 반영하지 못한다.

본 연구에서는 이러한 문제를 해결하기 위해 그래프 어텐션 네트워크[15]를 이용해 각 단계에서 만들어진 트리 정보를 디코더의 입력에 반영해 어절들을 표현해 보고자 한다.

디코더의 입력은 [12]의 연구와 같이 내부 스택을 이용한다. 디코더의 입력에 사용되는 어절들은 그래프 어텐션 네트워크를 통해 표현한다(d_t). 그래프 어텐션 네트워크는 3.2절에서 자세히 다룬다. 표현된 임베딩(d_t)은 LSTM의 입력으로 사용된다[Equation (3)].

$$\begin{aligned} S &= \{s_1, s_2, ..., s_M\} \\ d_t &= GAT(S) \\ h_t &= LSTM(d_t, h_{t-1}) \end{aligned} \tag{3}$$

스택-포인터 네트워크 모델에서 의존소를 찾는 방법은 Biaffine 어텐션[10]을 사용한다. 인코더(s_i)와 디코더(h_t)에서 생성한 표현들을 입력으로 하여 Equation (4)와 같이 어텐션을 계산한다. 그 중에서 가장 높은 값을 갖는 어절을 해당 어절의 의존소로 결정한다. 의존 구문 분석은 지배소와 의존소를 찾는 것뿐만 아니라 의존 관계도 분류해야 한다. 이를 위해 또 다른 Biaffine 어텐션을 이용한 분류 모델을 사용한다.

$$\begin{aligned} e_i^t &= h_t^T W s_i + U^T h_t + V^T s_i + b \\ p^t &= softmax(e^t) \end{aligned} \tag{4}$$

3.2 그래프 어텐션 네트워크를 이용한 부분 트리 표현

본 연구에서는 각 단계마다 만들어진 트리 정보를 반영하기 위해 그래프 어텐션 네트워크[15]를 채택한다. 이를 통해 출력된 어절 표현들은 디코더 LSTM의 입력으로 사용된다. 그래프 어텐션 네트워크는 인접한 노드들과의 어텐션을 계산해서 새로운 노드 표현을 생성한다(Fig. 3). Equation 5는 그래프 어텐션 네트워크의 수식이다. H는 멀티-헤드의 개수로 각 노드는 인접한 노드들과 H번 어텐션을 계산하고 이를 연쇄적(Concatenation)으로 결합한다. 네트워크의 각 노드(d)는 h번째 헤드의 파라미터 W^h를 곱하여 새로운 노드 표현($W^h d$)을 생성한다. 새롭게 생성한 노드 표현을 이용해 인접한 노드 d_i와 d_j 사이의 h번째 헤드의 어텐션(α_{ij}^h)을 계산하고 어텐션과 인접한 노드 표현을 곱한다. 어텐션이 반영된 인접한 노드들을 전부 더하여 각 노드의 표현을 새롭게 업데이트 한다. Equation (5)에서 N_i는 노드 i 자신과 그와 인접한 노드의 집합이다.

L은 그래프 어텐션 네트워크의 계층 수이다. L이 1일 경우에는 각 노드의 표현은 바로 인접한 노드들 사이의 어텐션만을 반영하여 계산한다. 하지만 L이 1보다 클 경우에는 각 노드의 표현이 네트워크의 레이어를 통과할 때마다 업데이트 되기 때문에 L+1차 정보까지 반영할 수 있다. Equation (5)에서 $d_{t,i}^l$는 t단계에서 l 개 레이어를 통과한 후의 d_i 노드의 표현이다.

$$\begin{aligned} g_{ij}^h &= f(A^{h^T}[W^h d_i \parallel W^h d_j]) \\ &(A^h \in R^{2*\dim_{out}}, W^h \in R^{\dim_{out} \times \dim_{in}}, f: leakyReLU) \\ \alpha_{ij}^h &= softmax_j(g_{ij}^h) = \frac{\exp(g_{ij}^h)}{\sum_{k \in N_i} \exp(g_{ik}^h)} \\ d_{t,i}^l &= \parallel_{h=1}^{H} \sigma\Big(\sum_{j \in N_i} \alpha_{ij}^h (W^{h,l} d_j^{l-1})\Big) \\ &(d^0 = \{s_1, ..., s_M\}, \sigma: elu) \end{aligned} \tag{5}$$

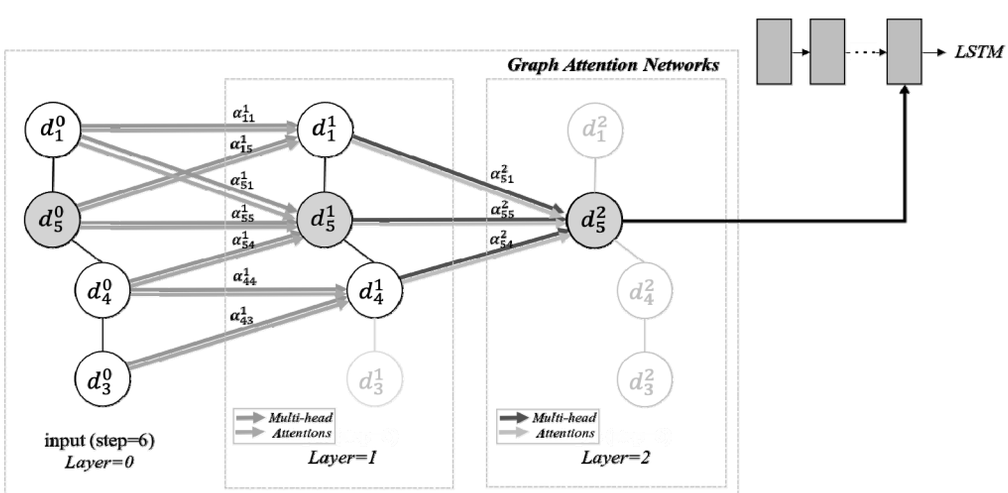

Fig. 3. Representation of Subtree using Graph Attention Networks

4. 코퍼스 정보 및 실험 환경

4.1 코퍼스 정보

본 연구의 실험에서는 2개의 코퍼스를 사용하였다. 첫째는 국립국어원[19]에서 배포한 세종 코퍼스로 구구조 구문 트리를 의존 구문 구조로 변환[20]하여 사용하였다. 세종 구문 분석 데이터는 학습용 53,842 문장과 평가용 5,817 문장으로 구성되어 있다(Table 1).

Table 1. Information of Sejong Corpus

Sejong Corpus (2010)	Number of Sentences	Average of Words	Average of Wordpieces
Train	53,842	11.19	31.22
Evaluation	5,817	9.93	28.02

둘째는 2020년에 국립국어원에서 공개한 모두의 말뭉치로 형태소 분석 말뭉치[21] 중 문어 버전과 구문 분석 말뭉치[22]를 사용하였다. 총 문장의 수는 149,500개이다. 모델 학습 및 평가를 위해 Table 2와 같이 학습용(119,500 문장) / 개발용(15,000 문장) / 평가용(15,000 문장) 데이터로 나누어 사용하였다[1].

Table 2. Information of Everyone's Corpus

NKL[2] (2020, ver 1.0)	Number of Sentences	Average of Words	Average of Wordpieces
Train	119,500	13.29	38.41
Validation	15,000	13.41	38.68
Evaluation	15,000	13.33	38.49

4.2 실험 환경

1) KorBERT

본 연구에서는 입력 문장의 단어 간 문맥 정보를 반영하기 위해 인코딩 단계에서 BERT를 사용했다. BERT 모델은 ETRI에서 공개한 KorBERT[3]를 사용했다. KorBERT는 신문기사와 백과사전 등 23GB 말뭉치로 사전 훈련된 언어 모델이다. 형태소 단위로 분석된 어절을 사용하기 위해 형태소 기반의 KorBERT를 채택하였다. KorBERT의 입력 단위는 형태소 기반의 워드피스이다. Table 3은 인코더의 입력을 표현한 예제이다. 인코더의 입력은 형태소 기반의 워드피스를 연속적으로 이어서 넣어준다. 본 연구에서는 "ROOT" 토큰을 KorBERT의 입력과 맞춰주기 위해 "[CLS]" 토큰으로 대체해 사용했다. KorBERT 모델에서 사용된 워드피스 개수는 30,349개이고 모델은 BERT-base[16]과 같다.

1) 이 사이트에서 나뉘어진 학습/개발/평가 데이터를 이용할 수 있음. (https://github.com/yseokchoi/DependencyCorpusOfEveryone)
2) National Institute of Korean Language
3) https://aiopen.etri.re.kr

Table 3. Inputs of KorBERT

Original Words	Morpheme-based Words	Morpheme-based WordPieces
[CLS]	-	[CLS]
대화는	대화/NNG 는/JX	대화/NNG_ 는/JX_
일상의	일상/NNG 의/JKG	일상/NNG_ 의/JKG_
관심사를	**관심사/NNG** 를/JKO	**관 심사/NNG_** 를/JKO_
규정한다	규정/NNG 하/XSV ㄴ다/EF	규정/NNG_ 하/XSV_ ㄴ다/EF_
[SEP]	-	[SEP]

2) 모델 및 학습 환경

본 연구에서 제안한 모델의 환경은 [5]의 연구에서 사용한 환경과 같다(Table 4). 모델 학습을 위한 환경은 Table 5와 같다.

Table 4. Parameters of Stack-pointer based Dependency Parser with Graph Attention Networks

Modules	Parameters	Values
Encoder	Mode	Bi-directional LSTM
	Number of layers	3
	Hidden dimension	256
Decoder	Model	Uni-directional LSTM
	Number of layers	2
	Hidden dimension	256
	Arc dimension	512
	Label dimension	128
Graph Attention Networks	Hidden dimension	512
	Number of attention heads	16

Table 5. Training Setup of the Proposed Dependency Parser

Parameters	Value
Dropout	0.1
Batch size	32
Optimizer	Adam
Initial learning rate of encoder	2e-5
Initial learning rate of decoder	1e-3
Beta1, Beta2	0.9, 0.998
Learning rate scheduler	Exponential LR Scheduler
Warm-up steps	40

Table 6. Parsing Performance of the Proposed Model on Sejong Corpus

Models		Unit of Word		Unit of Sentence	
		UAS	LAS	UCM	LCM
baseline	StackPtr(sibling) [5]	91.98	90.24	59.41	52.85
BERT Freeze	StackPtr(sibling) + BERT(Last)	93.70	91.80	64.60	56.56
	StackPtr(sibling) + BERT(Weight)	93.83	92.12	65.31	57.45
BERT Fine-tuning	StackPtr(sibling) + BERT	**94.41**	92.69	67.23	56.76
	StackPtr(grandpar,sibling) + BERT	94.36	92.62	66.99	59.12
	StackPtr + GAT(Layer=1)	94.37	92.65	67.17	59.45
	StackPtr + GAT(Layer=2)	94.39	**92.71**	**67.27**	**59.58**

Table 7. Comparison of the Proposed Model and Previous Studies

Models	UAS	LAS
BERT + LSTM + biaffine[6]	94.06	92.00
RoBERTa + biaffine[7]	94.32	92.40
BERT + GNN + biaffine[8]	**94.44**	92.55
StackPtr + GAT(Layer=2) (Our Model)	94.39	**92.71**

Table 8. Parsing Performance of the Proposed Model on Everyone's Corpus

Models	Unit of word		Unit of Sentence	
	UAS	LAS	UCM	LCM
StackPtr(sibling)+BERT (baseline)	93.46	91.34	49.62	39.85
StackPtr(grandpar,sibling) + BERT	93.51	91.32	49.83	39.87
StackPtr + GAT(Layer=1)	93.54	91.39	50.60	40.77
StackPtr + GAT(Layer=2)	**93.76**	**91.63**	**51.62**	**41.79**

5. 실험 결과

제안한 모델의 실험에 앞서 BERT 언어 모델을 추가한 실험을 진행했다. 의존 구문 분석기를 학습할 때 BERT 언어 모델의 파라미터를 업데이트를 하지 않은 경우(BERT Freeze)와 업데이트 하는 경우(BERT Fine-tuning)로 나누어 실험을 수행하였다. BERT Freeze는 BERT모델의 마지막 계층에서 생성한 표현을 인코더의 LSTM 입력에 넣어주는 방법(Last)과 BERT모델의 각 계층마다 생성한 표현을 학습이 가능한 가중치를 이용해 서로 더해주어 LSTM 입력에 넣어주는 방법(Weight)으로 나누어 실험을 수행하였다. BERT Fine-tuning은 구문 분석기를 학습할 때 BERT 모델의 파라미터도 같이 업데이트를 시켜준다.

Table 6은 세종 코퍼스를 이용해 모델을 학습하고 실험을 수행한 결과이다. 베이스라인 모델은 [5] 연구 모델에 대한 의존 구문 분석 실험 결과이다. BERT 언어 모델이 추가되면서 베이스라인보다 전체적으로 성능이 향상하는 것을 확인할 수 있다. 또한, BERT Freeze의 모델보다 BERT Fine-tuning 모델의 성능이 UAS 기준 0.5% 정도 향상되었다.

본 연구에서 제안한 모델은 그래프 어텐션 네트워크의 레이어를 2로 설정했을 때, UAS 기준을 제외하고는 제일 좋은 성능을 보였다. 이는 그래프 어텐션 네트워크가 트리에서 연결된 노드의 가중치를 이용하여 각각의 노드를 잘 표현하였고 더 나아가 계층이 깊어지면서 확장된 부분 트리를 디코더에 반영할 수 있었기 때문으로 판단된다.

Table 7은 사전 훈련된 언어 모델들을 사용한 의존 구문 분석 모델들에 대한 기존 실험 결과이다. 본 연구에서 제안한 모델은 UAS 점수가 낮지만 LAS 점수는 가장 높은 점수를 보여주었다.

Table 8은 제안한 모델을 모두의 말뭉치로 학습한 실험 결과이다. 실험은 BERT Fine-tuning에 대해서만 수행하였다. 본 연구에서 제안한 모델을 그래프 어텐션 네트워크의 계층을 2개 쌓았을 때 어절 단위의 정확도에서 UAS는 93.76, LAS는 91.64로 가장 높은 성능을 보여주었다. 문장 단위의 정확도에서도 UCM은 51.62, LCM은 41.79로 베이스라인 모델보다 약 2%의 높은 성능 향상을 보였다. 이는 제안한 모델의 그래프 어텐션 네트워크가 제공하는 부분 트리 정보가 전체 트리의 완성도를 강화시키는 효과가 있는 것으로 판단된다.

6. 결 론

본 연구에서는 스택-포인터 네트워크 모델에 그래프 어텐션 네트워크를 결합하여 한국어 의존 구문 분석기를 구현하였다. 제안한 모델은 각 단계마다 만들어진 트리 정보를 부분적으로 반영하기 위해 그래프 어텐션 네트워크를 사용하였

다. 그래프 어텐션 네트워크를 적용하였을 때 UAS에서는 점수가 좀 떨어졌지만 다른 평가 척도에서는 좋은 점수를 보여주었다. 특히 문장 단위의 트리 완성도에서는 다른 비교 모델들에 비해 좋은 성능을 보여주었는데 이는 디코딩 단계에서 각 단어의 의존소를 찾을 때, 앞서 만들어진 트리 정보를 반영하는 것이 문장 단위의 정확한 트리를 유도하는데 도움이 되기 때문으로 판단된다. 최근에 공개된 모두의 말뭉치 코퍼스에 대해서도 제안한 모델의 성능을 평가해 보았다. 그 결과 평가 데이터에 대해 UAS는 93.76, LAS는 91.63에 성능이 나왔으며, UCM은 51.77, LCM은 41.79의 성능으로 기존의 세종 코퍼스로 학습한 것보다는 다소 낮은 성능을 보였다. 이는 모두의 말뭉치 문장들이 세종 코퍼스의 문장들에 비해 평가 데이터 기준 약 3어절 이상 길기 때문으로 판단된다.

References

[1] S.-H. Na, J. Li, J.-H. Shin, and K. Kim, "Stack LSTMs with Recurrent Controllers for Korean Dependency Parsing," in *Proceedings of the 43th Annual Conference on KIISE*, pp.446-448, 2016.

[2] J.-W. Min and S.-H. Na, "SyntaxNet Models using Transition Based Recurrent Unit for Korean Dependency Parsing," in *Proceedings of the KIISE Korea Computer Congress*, pp.602-604, 2017.

[3] S.-H. Na, J. Li, J.-H. Shin, and K. Kim, "Deep Biaffine Attention for Korean Dependency Parsing," in *Proceedings of the KIISE Korea Computer Congress*, pp.584-586, 2017.

[4] C. Park, H. Hwang, C. Lee, and H. Kim, "Korean Dependency Parsing with Multi-layer Pointer Networks," in *Proceedings of the 29th Annual Conference on Human and Cognitive Language Technology*, pp.92-96, 2017.

[5] Y. Choi and K. J. Lee, "Korean Dependency Parser using Higher-order features and Stack-Pointer Networks," *Journal of KIISE*, Vol.46, No.7, pp.636-643, 2019.

[6] C. Park, C. Lee, J.-H. Lim, and H. Kim, "Korean Dependency Parsing with BERT," in *Proceedings of the KIISE Korea Computer Congress*, pp.530-532, 2019.

[7] J. Min, S.-H. Na, J.-H. Shin, and Y.-K. Kim, "RoBERTa for Korean Natural Language Processing: Named Entity Recognition, Sentiment Analysis, Dependency Parsing," in *Proceedings of the KIISE Korea Software Congress*, pp.407-409, 2019.

[8] J.-W. Min, S.-Y. Hong, Y.-H. Lee, and S.-H. Na, "Graph Neural Networks for Korean Dependency Parsing," in *Proceedings of the 31th Annual Conference on Human and Cognitive Language Technology*, pp.537-539, 2019.

[9] J.-H. Lim, Y. Bae, H. Kim, Y. Kim, and K.-C. Lee, "Korean Dependency Guidelines for Dependency Parsing and Exo-Brain Language Analysis Corpus," in *Proceedings of the 27th Annual Conference on Human and Cognitive Language Technology*, pp.234-239, 2015.

[10] T. Dozat and C.D. Manning, "Deep biaffine attention for neural dependency parsing," arXiv preprint arXiv:1611.01734, 2016.

[11] C. Dyer, M. Ballesteros, W. Ling, A. Matthews, and N.A. Smith, "Transition-based dependency parsing with stack long short-term memory," arXiv preprint arXiv:1505.08075, 2015.

[12] X. Ma, Z. Hu, J. Liu, N. Peng, G. Neubig, and E. Hovy, "Stack-pointer networks for dependency parsing," arXiv preprint arXiv:1805.01087, 2018.

[13] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *Advances in Neural Information Processing Systems*, pp.2692-2700, 2015.

[14] T. Ji, Y. Wu, and M. Lan, "Graph-based dependency parsing with graph neural networks," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp.2475-2485, 2019.

[15] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," arXiv preprint arXiv:1710.10903, 2017.

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

[17] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," arXiv preprint arXiv:1907.11692, 2019.

[18] Y. Wu, M. Schuster, Z. Chen, Q.V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, and K. Macherey, "Google's neural machine translation system: Bridging the gap between human and machine translation," arXiv preprint arXiv:1609.08144, 2016.

[19] CORPUS, "21st Century Sejong Project". 2010: The National Institute of the Korean Language.

[20] C. Lee, J. Kim, and J. Kim, "Korean Dependency Parsing using Deep Learning," in *Proceedings of the 26th Annual Conference on Human and Cognitive Language Technology*, pp.87-91, 2014.

[21] CORPUS. National Institute of the Korean Language Morphological Analysis Corpus (Version 1.0). 2020, [Internet] https://corpus.korean.go.kr/.

[22] CORPUS. National Institute of the Korean Language Parsing corpus (Version 1.0). 2020, [Internet] https://corpus.korean.go.kr/.

최 용 석

https://orcid.org/0000-0002-7889-8004
e-mail : yongseok.choi.92@gmail.com
2016년 충남대학교 정보통신공학과(학사)
2016년 ~ 2017년 충남대학교
전자전파정보통신공학과(석사)
2018년 ~ 현 재 충남대학교
전자전파정보통신공학과 박사과정
관심분야 : 자연언어처리, 정보검색, 기계학습, 인공지능

이 공 주

https://orcid.org/0000-0003-0025-4230
e-mail : kjoolee@cnu.ac.kr
1992년 서강대학교 전자계산학과(학사)
1994년 한국과학기술원 전산학과(공학석사)
1998년 한국과학기술원 전산학과(공학박사)
1998년 ~ 2003년 한국마이크로소프트(유)
연구원
2003년 이화여자대학교 컴퓨터학과 대우전임강사
2004년 경인여자대학 전산정보과 전임강사
2005년 ~ 현 재 충남대학교 정보통신공학과 교수
관심분야 : 자연언어처리, 기계번역, 정보검색, 정보추출