긴 문서를 위한 BERT 기반의 End-to-End 한국어 상호참조해결

조경빈*○, 정영준*, 이창기*, 류지희**, 임준호**

*강원대학교 빅데이터메디컬융합학과, **한국전자통신연구원
{jkb5570, kongjun, leeck}@kangwon.ac.kr, {chrisjihee, joonho.lim}@etri.re.kr

Korean End-to-End Coreference Resolution with BERT for Long Document

Kyeongbin Jo*○, Youngjun Jung*, Changki Lee*, Jihee Ryu**, Joonho Lim**
*Department of Big Data Medical Convergence, Kangwon National University
**Electronics and Telecommunications Research Institute

요 약

상호참조해결은 주어진 문서에서 상호참조해결 대상이 되는 멘션(mention)을 식별하고, 동일한 개체(entity)를 의미하는 멘션들을 찾아 그룹화하는 자연어처리 태스크이다. 최근 상호참조해결에서는 BERT를 이용하여 단어의 문맥 표현을 얻은 후, 멘션 탐지와 상호참조해결을 동시에 진행하는 end-to-end 모델이 주로 연구되었으나, 512 토큰 이상의 긴 문서를 처리하기 위해서는 512 토큰 이하로 문서를 분할하여 처리하기 때문에 길이가 긴 문서에 대해서는 상호참조해결 성능이 낮아지는 문제가 있다. 본 논문에서는 512 토큰 이상의 긴 문서를 위한 BERT 기반의 end-to-end 상호참조해결 모델을 제안한다. 본 모델은 긴 문서를 512 이하의 토큰으로 쪼개어 기존의 BERT에서 단어의 1차 문맥 표현을 얻은 후, 이들을 다시 연결하여 긴 문서의 Global Positional Encoding 또는 Embedding 값을 더한 후 Global BERT layer를 거쳐 단어의 최종 문맥 표현을 얻은 후, end-to-end 상호참조해결 모델을 적용한다. 실험 결과, 본 논문에서 제안한 모델이 기존 모델과 유사한 성능을 보이면서(테스트 셋에서 0.16% 성능 향상), GPU 메모리 사용량은 1.4배 감소하고 속도는 2.1배 향상되었다.

주제어: 상호참조해결, BERT, end-to-end

1. 서론

상호참조해결(coreference resolution)은 명사, 대명사, 명사구 등의 멘션(mention) 후보를 식별하고, 동일한 개체(entity)를 의미하는 멘션들을 찾아 그룹화(clustering) 하는 자연어처리 태스크이다. 최근 한국어 상호참조해결 연구는 포인터 네트워크(pointer network)를 사용하는 모델[1-4]과 멘션 탐지 태스크와 상호참조해결 태스크를 동시에 진행하는 end-to-end 상호참조해결 모델[5,6,7]가 주로 연구되었다.

구글에서 공개한 BERT(Bidirectional Encoder Representations from Transformer)[8]는 다양한 자연어처리 태스크에 적용되어 각 태스크의 성능을 향상시켰다. BERT는 트랜스포머(transformer)의 인코더(encoder) 부분으로 구성되며, BPE(Byte Pair Encoding)[9]를 사용하여 미등록어(unknown word) 문제를 해결하였으며, BPE를 적용한 대용량 말뭉치를 기반으로 사전 학습한다. 사전 학습을 위해 MLM(Masked Language Model)과 다음 문장 예측을 함께 학습하게 된다.

최근 한국어 상호참조해결 연구[4,7]에서는 BERT를 이용하여 단어의 문맥 표현을 얻었으나, 512 토큰 이상의 긴 문서를 처리하기 위해서는 512 토큰 이하로 문서를 분할하여 처리하기 때문에 길이가 긴 문서에 대해서는 상호참조해결 성능이 낮아지는 문제가 있다.

본 논문에서는 512 토큰 이상의 긴 문서를 위한 BERT 기반의 end-to-end 상호참조해결 모델을 제안한다. 이를 위해 512 토큰 이상의 긴 문서를 512 이하의 토큰으로 쪼개어 기존의 Local BERT에서 local attention 정보를 이용하여 단어의 1차 문맥 표현을 얻고, 이들을 다시 연결(concatenation)하여 쪼개기 전의 긴 문서의 Global Positional Encoding 또는 Embedding 값을 더한 후, Global BERT layer를 거쳐 전체 문서의 global attention 정보를 이용하여 단어의 최종 문맥 표현을 얻은 후, end-to-end 상호참조해결 모델을 적용한다. 실험 결과, 본 논문에서 제안한 모델이 기존 모델과 유사한 성능을 보이면서(테스트 셋에서 0.16% 성능 향상), GPU 메모리 사용량은 1.4배 감소하고 속도는 2.1배 향상되었다.

2. 관련 연구

최근 한국어 상호참조해결 연구는 포인터 네트워크를 사용하는 모델[1-4]과 멘션 탐지 태스크와 상호참조해결 태스크를 동시에 진행하는 end-to-end 모델[5,7]이 연구되었다. [1]은 RNN encoder-decoder를 기반으로 문맥 정

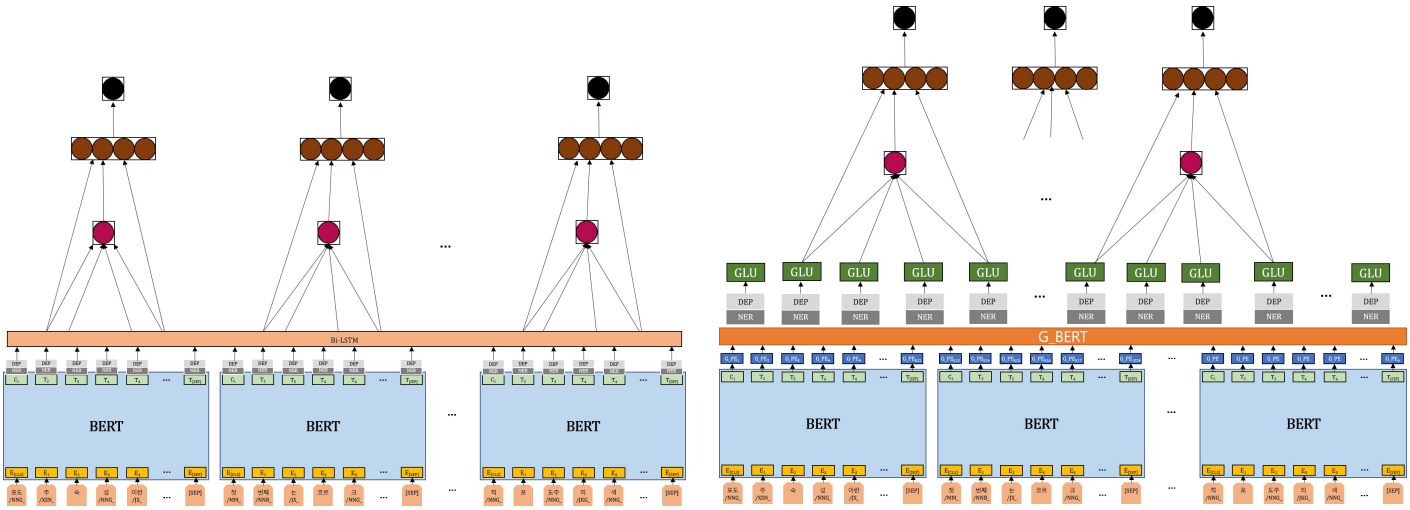

그림 1 BERT를 이용한 end-to-end 상호참조해결 모델 멘션 스코어 구조, 기존 BERT에 개체명 자질, 의존구문분석 자질을 연결하여 bi-LSTM 레이어를 추가한 모델[7](왼쪽), 왼쪽 모델에서 bi-LSTM를 제거하고 G_BERT와 Global Positional Embedding, GLU가 추가된 모델(오른쪽)

보를 인코딩(encoding)하고 포인터 네트워크를 이용하여 참조관계를 밝히는 시도를 하였다. [2]는 셀프매칭 어텐션(self-matching attention)을 기반으로 포인터 네트워크를 적용한 상호참조해결 모델을 제안하였다. [3]은 양방향 언어 모델을 사전 학습한 ELMo[10] 표현과 음절 표현을 학습하는 CNN(Convolution Neural Network)을 적용한 포인터 네트워크 모델을 제안하였다. [4]는 포인터 네트워크에 BERT와 Deep Biaffine을 이용하여 멘션 탐지 태스크와 상호참조해결 태스크를 한 번에 해결하는 모델을 제안하였으며, 한국어 상호참조해결에 많은 성능 향상을 보였다.

[5,6,7]의 연구는 가능한 모든 스팬을 멘션 후보로 간주하여 멘션 탐지 태스크와 상호참조해결 태스크를 동시에 해결하는 end-to-end 모델이다. [5]는 기존 end-to-end 모델에 문장 수준에서만 상호참조가 되는 일관성 문제(consistency errors)를 해결하기 위해 고차 추론(higher-order)를 적용하고, 고차 추론의 복잡도를 줄이기 위해 coarse-to-fine도 적용하였다. [6]은 end-to-end 모델에 BERT를 추가하여 영어 상호참조해결에서 많은 성능 향상을 보였으며, 토큰의 개수가 512 이상인 문서를 BERT에 입력하기 위하여 문서를 정해진 길이의 세그먼트 단위로 나누어 하나의 배치로 병렬로 수행하여 길이가 긴 문서에 대해 성능 향상을 보였다.

[7]는 [6]의 모델에 한국어의 특성을 반영하기 위해 개체명 자질과 의존 구문 분석 자질을 적용하고, bi-LSTM 레이어를 추가하여 BERT 표현과 문맥 정보에 대한 히든 스테이트(hidden state)를 만들어 한국어 상호참조해결을 수행하는 모델이다. 하나의 문서를 512크기의 세그먼트 단위로 나누어 하나의 배치로 병렬로 수행하여 문맥 표현을 얻은 후 이를 다시 연결하여 bi-LSTM의 입력으로 주어 전체 문맥 정보를 모델링한다.

3. 긴 문서를 위한 BERT 기반 End-to-End 상호참조해결 모델

End-to-end 상호참조해결 모델은 선행사와 멘션 쌍이 상호참조될 조건부 확률 분포 $P(y_1,\ldots,y_N|D)$를 학습하는 것이 목표이다. 여기에서 D는 입력 문서, y_i는 멘션 후보, N은 D에서의 스팬 후보 수이다. D의 입력 토큰 수를 T라 할 때 스팬 후보 수 $N = T(T+1)/2$이다. End-to-end 상호참조해결 모델은 그림 1과 같이 스팬 표현(span representations)과 멘션 스코어를 통해 멘션 후보들(k개)을 결정하는 단계, 추출된 멘션 후보들과 선행사 후보(c개)간의 상호참조해결을 결정하는 단계로 나뉜다. 이 과정에서 k, c는 하이퍼 파라미터로 조절 가능한 값이다.

기존에 제안한 BERT 기반의 end-to-end 상호참조해결 모델[7]는 BERT를 이용하여 단어의 문맥 표현(contextual representation)을 얻고 한국어 특성을 반영하기 위하여 개체명 자질과 의존 구문 분석 자질을 추가하여 bi-LSTM 레이어를 통해 단어의 최종 문맥 표현을 얻었다.

본 논문에서 제안하는 긴 문서를 위한 BERT 기반의 end-to-end 상호참조해결 모델은 긴 문서를 512 이하의 토큰으로 쪼개어 기존의 Local BERT에서 local attention 정보를 이용하여 단어의 1차 문맥 표현을 얻고, 이들을 다시 연결하여 쪼개기 전의 긴 문서의 Global Positional Encoding 또는 Embedding 값을 더한 후 Global BERT layer를 거쳐 전체 문서의 global attention 정보를 이용하여 단어의 최종 문맥 표현을 얻는다.

3.1 스팬 표현과 멘션 스코어

입력 토큰에 대한 문맥 표현 h_t는 식(1-4)와 같이 계산된다.

$$z_t = \text{BERT}(x_t) \oplus \text{G\_PE}(x_t) \quad (1)$$

$$v_t = [\text{G\_BERT}(z_t); ner(x_t); dep(x_t)] \quad (2)$$

$$GLU(x) = x \otimes \sigma(FFNN(x)) \quad (3)$$

$$h_t = GLU(v_t) \quad (4)$$

형태소 단위에 BPE가 적용된 입력열 토큰 $X = \{x_1, \ldots, x_T\}$를 512 이하의 토큰으로 쪼개어 기존의 Local BERT를 이용하여 얻은 $\text{BERT}(x_t)$에 쪼개기 전의 긴 문서의 Global Positional Encoding 또는 Embedding(G_PE)을 element wise sum을 수행하여 z_t를 계산한다. 이렇게 구한 z_t를 Global BERT layer에 입력으로 주어 $G_BERT(z_t)$를 얻은 후, 의존 구문 분석 자질(dep)과 개체명 자질(ner)을 연결한 벡터 v_t를 GLU(Gated Linear Unit)의 입력으로 주어 h_t를 구한다. h_t는 스팬 표현 g_i를 만들기 위해 사용되며, 식(8)과 같이 스팬의 시작을 나타내는 $h_{START}(i)$, 끝을 나타내는 $h_{END}(i)$, 스팬의 중심어 표현 $\hat{h}_t$, 자질벡터 $\phi(i)$를 연결하여 g_i를 얻는다.

$$a_t = w_a \cdot FFNN_a(h_t) \quad (5)$$

$$a_{i,t} = \frac{\exp(a_k)}{\sum_{k=START(i)}^{END(i)} \exp(a_k)} \quad (6)$$

$$\hat{h}_t = \sum_{t=START(i)}^{END(i)} a_{i,t} \cdot x_t \quad (7)$$

$$g_i = [h_{START(i)}, h_{END(i)}, \hat{h}_t, \phi(i)] \quad (8)$$

$$s_m(i) = w_m \cdot FFNN_m(g_i) \quad (9)$$

위 식에서 $span_i$의 시작과 끝 위치는 각각 $START(i)$, $END(i)$로 표현한다. 식(5-7)은 중심어 표현 $\hat{h}_t$를 구하는 과정이며, $span_i$의 모든 토큰들에 대한 가중치 합을 수행한 값이다. 가중치 a_t는 FFNN(Feed-Forward Neural Network)으로 계산된다. 계산된 스팬 표현 g_i는 이후, 멘션 스코어 $s_m(i)$를 구하는데 사용되며, 멘션 스코어 기준 상위 k개를 멘션 후보로 정한다.

3.2 상호참조해결 스코어와 고차 추론

3.1절에서 구해진 멘션 후보들로, 선행사 여부를 판별하는 선행사 스코어 s_c, s_a와, 최종적으로 상호참조 여부를 결정하는 상호참조해결 스코어 $s(i,j)$를 구하며, 식은 아래와 같다.

$$s_c(i,j) = g_i^T W_c g_j \quad (10)$$

$$s_a(i,j) = w_a \cdot FFNN_a([g_i, g_j, g_i \circ g_j, \phi(i,j)]) \quad (11)$$

$$s(i,j) = \begin{cases} 0 & j = \epsilon \\ s_m(i) + s_m(j) + s_a(i,j) + s_c(i,j) & j \neq \epsilon \end{cases} \quad (12)$$

위 식에서 i와 $j(1 \le j \le i-1)$는 현재 멘션과 선행사의 인덱스이며, g_i와 g_j는 i와 j멘션의 스팬 표현을 나타낸다. 선행사 스코어 $s_a(i,j)$와 $s_c(i,j)$는 g_j와 g_i의 선행사인지 여부를 나타내는 스코어이다. 그 중, $s_a(i,j)$는 멘션, 선행사 각각의 스팬 표현과 element wise의 결과인 $g_i \circ g_j$, 두 스팬의 거리 자질인 $\phi(i,j)$를 연결하여 FFNN을 수행한다. 이 과정의 시간 복잡도는 $O(n^3)$이며, [5,7]에서는 계산 비용을 줄이기 위하여 bilinear 연산인 $s_c(i,j)$를 도입하여 coarse-to-fine을 진행 후, 상위 c개에 대하여 $s_a(i,j)$ 선행사 스코어를 계산한다. 마지막으로, 식(12)와 같이 멘션 스코어와 선행사 스코어를 합하여 상호참조해결 스코어를 계산한다. 또한 [5,7]는 문장 수준에서는 상호참조해결이 되는 것처럼 보이지만, 전체 문서로 보았을 때 상호참조해결이 안되는 일관성 문제를 해결하기 위하여 고차 추론을 적용하며, 식은 아래와 같다.

$$P_n(y_i) = \frac{e^{s(g_i^n, g_y^n)}}{\sum_{y \in Y(i)} e^{s(g_i^n, g_y^n)}} \quad (13)$$

$$a_i^n = \sum_{y \in Y(i)} P_n(y_i) \cdot g_{y_i}^n \quad (14)$$

$$f_i^n = \sigma(W_f[g_i^n, a_i^n]) \quad (15)$$

$$g_i^{n+1} = f_i^n \circ g_i^n + (1 - f_i^n) \circ a_i^n \quad (16)$$

고차 추론은 n번 반복하며, 현재 멘션의 선행사들에 대한 얼라인먼트 스코어 a_i^n와 현재 멘션 정보에 대한 게이트 벡터 f_i^n를 사용하여 스팬 표현을 업데이트 해준다. 모델의 출력은 상호참조해결 스코어가 가장 높은 k개의 (멘션, 선행사) 쌍이다.

4. 실험 및 결과

본 논문에서 제안한 길이가 긴 문서를 위한 상호참조해결 모델을 실험하기 위하여 평균 문서의 길이가 긴 ETRI WIKI 도메인 상호참조해결 데이터 셋을 이용하였다. 데이터 셋의 평균 문서 길이는 26 문장, 1704 음절이며, 데이터 셋은 학습 데이터(train) 891 문서, 개발 데이터(dev) 50 문서, 평가 데이터(test) 50 문서로 구성된다. 상호참조해결의 성능 측정을 위하여 중심어 경계(head boundary)를 기준으로 MUC, B^3, CEAF-e, CoNLL F1[11]을 사용하였다.

본 논문에서 제안한 모델의 Local BERT를 위해 ETRI의 KorBERT(BERT-base)를 사용하였다. 상호참조해결 모델 학습을 위한 하이퍼 파라미터는 다음과 같다. BERT fine-turning 학습률(learning rate)은 1e-05, end-to-end 모델 학습률은 2e-4, FFNN은 드롭아웃(dropout) 0.2, 히든 레이어 차원수 100, 스택수는 2, 고차 추론 반복 횟수는 2로 설정하였다. 계산량을 위한 하이퍼 파라미터는 선행사 후보 수 50, 최대 멘션 후보 수는 600, 최대 스팬 길이 70, 멘션 스코어 pruning 비율은 0.4로 설정하였다. 학습을 위하여 Adam을 사용하였고, 미니 배치 크기는 1이다. 실험에 사용된 GPU는 RTX TITAN 24GB이다.

본 논문에서 제안한 모델의 Global Positional Encoding/Embedding 및 Global BERT layer의 효과를 검증

하기 위해 다음의 실험들을 진행하였다.

4.1 Global/Local BERT layer

Global BERT layer(G_BERT) 실험은 그림 1의 오른쪽 모델에서 Global BERT layer의 수와 Local BERT layer의 수를 변경하여 실험을 진행하였다. Global BERT layer의 초기값으로는 KorBERT의 마지막 레이어의 가중치를 사용하였다. 예를 들어, 레이어를 3개를 쓴 경우에는 KorBERT 레이어 10, 11, 12을 초기 가중치로 사용하였다.

표 1는 Local BERT 레이어의 수는 12로 고정하였고, Global BERT의 레이어 수의 변경에 따른 모델의 성능 비교 결과이다. 레이어를 1개만 사용하였을 경우 테스트 셋에서 67.22%로 성능이 가장 좋았으며, 레이어를 추가할수록 성능이 하락하였다.

표 2는 Local BERT 레이어와 Global BERT 레이어 수에 따른 모델의 성능 비교 결과이다. Local BERT 레이어 수 11개, Global BERT 레이어 수 1개를 사용하였을 때 성능이 가장 좋았으며, 표 1의 실험과 비슷하게 Global BERT 레이어를 추가할수록 성능이 하락하였다.

4.2 Global Positional Encoding/Embedding

Global Positional Encoding 및 Embedding 실험은 그림 1의 오른쪽 모델에서 G_PE 부분만 교체하면서 실험을 하였고, 4.1절의 실험에서 가장 좋은 성능을 보인 Local BERT 레이어 11개, Global BERT 레이어 1개를 사용하였다. Global Positional Encoding(G_P_Encoding)은 트랜스포머 모델에서 사용한 Positional Encoding과 동일한 식(17,18)을 사용하여 긴 문서의 Global Positional Encoding 값을 계산하였다.

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}}) \quad (17)$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}}) \quad (18)$$

Global Positional Embedding은 BERT의 Positional Embedding과 동일하게 구현하였고, 긴 문서의 토큰 수를 고려하여 Positional Embedding 크기를 4096으로 고정하여 실험을 진행하였으며, Positional Embedding의 초기값은 3가지 방법으로 초기화 하였다. 첫 번째 방법은 KorBERT의 Positional Embedding(512 크기) 가중치를 8번 반복해서 연결하여 초기화를 하였고(G_P_Embedding_tile), 두 번째 방법은 KorBERT의 Positional Embedding의 크기만큼 초기화를 진행한 후 남은 가중치에 대해서는 KorBERT의 Positional Embedding의 마지막 Embedding인 512번 Embedding의 가중치를 반복해서 연결하여 초기화하였고(G_P_Embedding_end), 마지막 세번째 방법은 랜덤하게 초기값을 주었다(G_P_Embedding_random).

표 3은 Global Positional Encoding 및 Embedding을 적용한 모델에 대한 성능 비교 결과이다. 실험 결과, KorBERT의 Positonal Embedding을 8번 반복해서 연결하여 초기화한 경우(G_P_Embedding_tile)가 개발 셋에서 68.54%, 테스트 셋에서 67.72% 로 가장 높은 성능을 보였다.

표 1 Global BERT 레이어 수에 따른 성능 비교

#Layer	MUC	B^3	CEAFe	Test F1
3	72.89	63.86	59.48	65.57
2	73.42	64.41	61.00	66.28
1	**74.04**	**65.46**	**62.16**	**67.22**

표 2 Global/Local BERT 레이어 수에 따른 성능 비교

#Layer	MUC	B^3	CEAFe	Test F1
L_12/G_1	74.04	65.46	62.16	67.22
L_11/G_1	**74.62**	**66.01**	**62.54**	**67.72**
L_10/G_2	72.98	63.95	59.93	65.62
L_9/G_3	72.35	63.24	59.35	64.98

표 3 Positional Encoding/Embedding 성능 비교

DEV				
Model	**MUC**	**B^3**	**CEAFe**	**F1**
G_P_Encoding	74.55	65.71	61.83	67.36
G_P_Embedding_random	75.00	66.57	63.42	68.33
G_P_Embedding_end	74.86	66.14	63.56	68.19
G_P_Embedding_tile	**75.21**	**66.76**	**63.65**	**68.54**
TEST				
Model	**MUC**	**B^3**	**CEAFe**	**F1**
G_P_Encoding	73.55	64.50	60.28	66.11
G_P_Embedding_random	73.88	65.37	61.91	67.05
G_P_Embedding_end	74.44	65.90	62.08	67.47
G_P_Embedding_tile	**74.62**	**66.01**	**62.54**	**67.72**

표 4 레이어 추가에 따른 성능 비교

Model	MUC	B^3	CEAFe	Test F1
Baseline[7]	74.50	65.84	62.34	67.56
Higher-order e2e-coref+BERT+G_BERT	74.22	65.83	62.67	67.57
+GLU	74.34	66.28	62.39	67.67
+Positional Embedding	**74.62**	**66.01**	**62.54**	**67.72**

표 5 모델에 따른 상호참조해결 속도 및 메모리 비교

Model	100 epochs Learning Time(hours)	GPU memory (GB)	50 docs perform time (seconds)
Baseline[7]	20	23	14.76
Our_model	**7**	**16**	**6.96**

표 4는 기존 BERT 기반의 end-to-end 상호참조해결 모델[7]에 bi-LSTM을 제거하고, 순차적으로 G_BERT, GLU, Positional Embedding을 추가한 성능을 비교한 결과이다. GLU, Positional Embedding을 추가함에 따라 성능이 추가적으로 오르며, 최종적으로 기존 모델[7]에 비해서 성능이 0.16% 향상되었다.

표 6 모델에 따른 실험 결과 예제 비교

Baseline[7]
[[에리얼과]^{0} [플라운더는]^{1}]^{2} [[에릭]^{3} 왕자의]^{4} 생일 축하연을 보기 위해 수면으로 올라가며, 거기서 [[에릭을]^{4} 본 에리얼은]^{0} [그에게]^{4} 한눈에 반한다. 그러나 축하연은 갑자기 몰아닥친 폭풍으로 중단되고, 배는 가라앉는다. [에릭은]^{3} 익사할 뻔했으나 [에리얼이]^{0} 구출하여 살아남고 해변에 눕혀진다. [에리얼은]^{0} [왕자에게]^{4} 향하는 [자신의]^{0} 마음을 노래하지만, [에릭이]^{3} 깨었을 때 [에리얼은]^{0} 이미 물속으로 숨긴 뒤였다.
Our_model
[[에리얼과]^{0} [플라운더는]^{1}]^{2} [[에릭]^{4} 왕자의]^{4} 생일 축하연을 보기 위해 수면으로 올라가며, 거기서 [[에릭을]^{4} 본 에리얼은]^{0} [그에게]^{4} 한눈에 반한다. 그러나 축하연은 갑자기 몰아닥친 폭풍으로 중단되고, 배는 가라앉는다. [에릭은]^{4} 익사할 뻔했으나 [에리얼이]^{0} 구출하여 살아남고 해변에 눕혀진다. [에리얼은]^{0} [왕자에게]^{4} 향하는 [자신의]^{0} 마음을 노래하지만, [에릭이]^{4} 깨었을 때 [에리얼은]^{0} 이미 물속으로 숨긴 뒤였다.

표 5는 기존 모델[7]에 비해서 본 논문에서 제안한 모델이 메모리 사용량과 속도 면에서 우수하다는 것을 보여준다. G_BERT를 적용한 모델의 경우 100 에폭 학습하는 데에 걸리는 시간은 7시간으로 기존 모델 대비 약 2.8배 향상된 것을 볼 수 있고, GPU 메모리 사용량은 23GB에서 16GB로 약 1.4배 감소하였고, 50문서를 수행하는 데에 걸리는 시간은 6.96초로 약 2.1배 향상되었다. 이는 기존 모델에서 전체 문서의 정보를 통합하기 위해 사용했던 bi-LSTM 대신에 self-attention(Global BERT layer)을 사용하여 GPU 메모리 사용량을 줄이고 GPU의 병렬성을 활용하여 속도를 향상시킨 것으로 생각된다.

4.3 실험 결과 예제

표 6은 본 논문에서 제안한 모델에서의 상호참조해결 추론 결과와 Baseline[7]의 상호참조해결 추론 결과이다. 사용된 예제는 25 문장, 1419 음절로 구성되어있으며, 그 중 일부를 추출하였다. 가시성을 위해 모든 멘션에 대해 표기하지 않고 인물에 대해서만 표기하였다. 본 논문에서 제안한 모델에서는 "에릭"과 왕자를 같은 엔티티로 묶었지만, Baseline[7]에서는 서로 다른 것으로 구분한 것을 보여주고 있다. 실제로는 같은 것을 지칭하지만 Baseline[7]에서는 첫 번째 문장에서의 "그에게"가 "에릭"이 아닌 왕자만을 가리키는 문제가 발생하고 있다. 이는 세그먼트 사이의 Global한 정보를 제대로 인코딩하지 못했기 때문으로 보인다.

5. 결론

본 논문에서는 BERT 기반의 end-to-end 상호참조해결 모델에서 긴 문서를 효율적으로 다루기 위해 기존 모델에서 사용한 bi-LSTM을 Global BERT layer(G_BERT)로 교체하고 Global Positional Embedding과 자질 추가를 위한 Gated Linear Unit(GLU)을 도입한 모델을 제안하였다. 실험 결과, 본 논문에서 제안한 모델이 기존 모델과 유사한 성능을 보이면서(테스트 셋에서 0.16% 성능 향상), GPU 메모리 사용량은 1.4배 감소하고 속도는 2.1배 향상되었다.

감사의 글

이 논문은 2021년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2013-2-00131, 휴먼 지식증강 서비스를 위한 지능진화형 Wise QA 플랫폼 기술 개발).

참고문헌

[1] 박천음, 이창기, "포인터 네트워크를 이용한 한국어 대명사 상호참조해결," *정보과학회논문지*, 제44권, 제5호, 2017.

[2] 박천음, 이창기, 김현기, "셀프 매칭 어텐션 기반 포인터 네트워크를 이용한 한국어 상호참조해결," *2017년 한국소프트웨어종합학술대회 논문집*, 2017.

[3] 박천음, 이창기, 류지희, 김현기, "문맥 표현과 음절 표현 기반 포인터 네트워크를 이용한 한국어 상호참조해결," *제30회 한글 및 한국어 정보처리 학술대회 논문집*, 2018.

[4] 박천음, 김기훈, 이창기, 임준호, 류지희, 김현기, "BERT기반 Deep Biaffine을 이용한 한국어 상호참조해결," *2019년 한국컴퓨터종합학술대회 논문집*, 2019.

[5] 김기훈, 박천음, 이창기, 김현기, "고차 추론을 이용한 한국어 End-to-End 신경망 기반 상호참조해결," *정보과학회 컴퓨팅의 실제 논문지*, 제26권, 제5호, 2020.

[6] Mandar Joshi, Omer Levy, Daniel S. Weld, Luke Zettlemoyer, "BERT for Coreference Resolution: Baselines and Analysis," *arXiv preprint arXiv:1908.09091*, 2019

[7] 김기훈, 박천음, 이창기, 김현기, "BERT 기반 End-to-end 신경망을 이용한 한국어 상호참조해결," *정보과학회논문지*, 제47권, 제10호, 2020.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[9] Rico Sennrich, Barry Haddow, Alexandra Birch, " Neural Machine Translation of Rare Words with Subword Units," *ACL*, 2016.

[10] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, Luke Zettlemoyer, "Deep contextualized word representations," *NAACL*, 2018.

[11] Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, "Deterministic Coreference Resolution Based on Entity-Centric, Precision-Ranked Rules," *Computational Linguistics*, Volume 39, Issue 4, 2013.