포인터 네트워크를 이용한 멘션탐지

박천음, 이창기
강원대학교 컴퓨터과학과
{parkce, leeck}@kangwon.ac.kr

Mention Detection using Pointer Networks

Cheoneum Park, Changki Lee
Kangwon National University

요 약

멘션(mention)은 명사 또는 명사구를 중심어로 가지며, 수식어를 포함하여 어떤 의미를 정의하는 구(chunk)를 구성한다. 문장 내에서 멘션을 추출하는 것을 멘션탐지라 한다. 멘션들 중에서 서로 같은 의미의 멘션들을 찾아내는 것을 상호참조해결이라 한다. 포인터 네트워크는 RNN encoder-decoder 모델을 기반으로, 주어진 입력 열에 대응되는 위치를 출력 결과로 갖는 모델이다. 본 논문에서는 멘션탐지에 포인터 네트워크를 이용할 것을 제안한다. 멘션탐지에 포인터 네트워크를 적용하면 기존의 sequence labeling 문제로는 해결할 수 없었던 중첩된 멘션탐지 문제를 해결할 수 있다. 실험 결과, 본 논문에서 제안한 멘션탐지의 성능이 규칙기반 보다 8%이상 높은 F1 80.75%를 보였으며, 이를 이용한 상호참조해결 성능이 CoNLL F1 52.69%로 규칙기반 멘션탐지를 이용한 상호참조해결에 비하여 2.29% 더 좋은 성능을 보였다.

주제어: 멘션탐지, 경계탐지, 포인터 네트워크, 상호참조해결

1. 서론

상호참조해결(Coreference resolution)은 문서 내에 등장한 임의의 단어들 간(즉, 현재 위치한 단어와 이전에 언급됐던 단어(선행사))에 서로 같은 개체를 가리키는 단어들의 관계를 명확하게 연관짓는 자연어처리 문제이다. 상호참조해결은 멘션(mention)을 기반으로 하며, 문서 내에서 멘션을 추출하는 방법을 멘션탐지(Mention detection)이라 한다. 본 논문에서는 RNN (Recurrent Neural Networks)의 확장 모델인 포인터 네트워크(Pointer Networks)를 이용하여 멘션탐지를 수행한다.

포인터 네트워크는 RNN Encoder-decoder 모델[1]을 기반으로 주어진 입력 열에 대응되는 위치를 출력 결과로 갖는 모델이다. 이 모델은 어텐션 메커니즘(Attention mechanism)[2]을 적용하여 가변적인 출력 클래스(입력 열의 길이가 가변적)에 대한 문제를 보다 더 정확하게 해결할 수 있다.

본 논문에서 제안한 포인터 네트워크를 이용한 멘션탐지는 기존의 sequence labeling 문제로는 해결할 수 없었던 중첩된 멘션탐지 문제를 해결할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서 관련 연구에 대하여 소개하고, 3장에서 포인터 네트워크에 대하여 설명한다. 4장에서는 본 논문에서 제안하는 방법인 포인터 네트워크를 이용한 멘션탐지 방법에 대하여 설명하고, 5장에서 실험을 보이며 결과를 분석한다. 6장을 마지막으로 본 논문의 결론 및 향후 연구 계획을 기술한다.

2. 관련 연구

멘션은 상호참조해결의 대상이기 때문에 멘션이 많을수록 상호참조해결의 재현율(recall)이 향상되는 효과를 볼 수 있다. 기존의 멘션탐지는 규칙기반[5]과 통계기반[6] 방법을 이용하여 수행되었다.

규칙기반 멘션탐지는 의존트리(dependency tree)에 기반 하여 명사나 명사구를 중심어(head)로 추출하였으며, 중심어와 수식어에 따라 멘션 경계를 정의하였다. 규칙기반의 경우에는 일반적인 단일 명사 또는 복합 명사, 짧은 수식어를 가지는 중심어 등과 같이 비교적 짧은 명사구의 경계를 정확하게 정의하여 올바른 멘션으로 추출할 수 있다. 반면에, 수식어가 길어지거나, 복문의 경우에는 정확한 경계를 갖는 멘션을 추출하기 어려우며, 이에 따라 멘션의 크로스, 수식어 정보 손실, 오류 누적 등의 문제가 발생할 수 있다[7].

위와 같은 문제를 해결하고자 [6]에서는 딥 러닝(Bidirectional LSTM-CRF 모델 등)을 이용한 통계기반 멘션탐지를 제안하였고, 긴 멘션 경계를 갖는 멘션을 추출하는데 있어서 규칙기반 보다 좋은 성능을 보였다. 그러나 멘션탐지의 경우, 문장 내에서 다수의 멘션이 중첩되는 문제로, 모든 멘션에 대한 통계기반 방법을 적용할 수 없기 때문에 문장에서 가장 긴 멘션 경계를 갖는 멘션에 대해서만 학습을 수행하고, 멘션을 추출하였다.

본 논문에서 제안하는 포인터 네트워크를 이용한 멘션탐지 방법은 기존 통계기반의 한계였던 중첩된 멘션의 멘션탐지가 가능하다.

3. 포인터 네트워크

포인터 네트워크는 입력 열에 대응되는 위치를 출력 결과로 갖는 RNN의 확장된 모델이다. 이 모델은 어텐션 메커니즘을 사용하여 가변길이 출력 클래스에 대하여 학습이 가능하다. 포인터 네트워크는 인코더(encoder)의 hidden state를 생성하고, 이와 함께 현재까지 생성한 디코더(decoder)의 hidden state를 입력으로 하여 입력 열 중 어느 위치를 주의하여 봐야 할지에 대한 어텐션 가중치(attention weight)를 학습한다[8].

본 논문에서는 인코더를 위하여 bidirectional Gated Recurrent Unit(bi-GRU)[1]를 사용하며, 수식은 아래와 같다.

$$\vec{h}_s = f_{GRU}(E(x_s), \vec{h}_{s-1})$$
$$\overleftarrow{h}_s = f_{GRU}(E(x_s), \overleftarrow{h}_{s+1})$$
$$\overleftrightarrow{h}_s = [\vec{h}_s, \overleftarrow{h}_s]$$

여기서 Forward, Backward Network는 $\vec{h}_s$와 $\overleftarrow{h}_s$이며, $E(x_s)$는 입력열의 s 번째 단어의 단어표현(word embedding)이다. $\overleftrightarrow{h}_s$는 $\vec{h}_s$와 $\overleftarrow{h}_s$를 연결(concatenate)한 것이다.

$$x_t = \overleftrightarrow{h}_{y_{t-1}}$$
$$h_t = f_{GRU}(x_t, h_{t-1})$$
$$a_t(s) = \frac{\exp\left(score\left(h_t, \overleftrightarrow{h}_s\right)\right)}{\sum_s \exp\left(score\left(h_t, \overleftrightarrow{h}_s\right)\right)}$$
$$score\left(h_t, \overleftrightarrow{h}_s\right) = \begin{cases} v_t^T \tanh\left(W_a\left[h_t; \overleftrightarrow{h}_s\right]\right), & concat \\ v_t^T \tanh\left(W_a\left[h_t; x_t; \overleftrightarrow{h}_s\right]\right), & concat2 \end{cases}$$
$$y_t = \text{argmax}(a_t(s'))$$

x_t는 현재 디코더의 입력으로, 이전 디코더의 출력 결과 위치에 해당하는 인코더의 hidden state이다. h_t는 x_t와 디코더의 이전 hidden state를 입력으로 하는 디코더의 hidden state이다. $a_t(s)$는 어텐션 가중치로, $score(h_t, \overleftrightarrow{h}_s)$ 함수의 결과 벡터에 ***softmax***를 이용하여 정규화한 값이며, 이 값으로 입력 열에 대응되는 위치를 결정한다. $score(h_t, \overleftrightarrow{h}_s)$ 함수는 ***concat***과 ***concat2*** 두 가지 방법 중 하나로 얼라인먼트 스코어(alignment score)를 계산하는데, 먼저 ***concat***은 h_t와 $\overleftrightarrow{h}_s$를 연결하여 점수를 계산하는 방법이고, ***concat2***는 h_t와 x_t, $\overleftrightarrow{h}_s$를 연결하여 점수를 계산하는 방법이다. [그림 1]은 $score(h_t, \overleftrightarrow{h}_s)$ 함수의 ***concat***과 ***concat2***에 대한 포인터 네트워크 구조를 보인다. 본 논문에서는 beam search를 이용하여 최적의 출력 열을 계산한다.

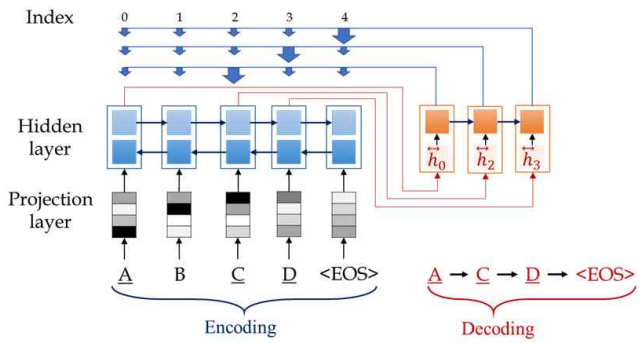

그림 1. $score$ 함수를 이용한 포인터 네트워크

[그림 1]은 입력열 [A, B, C, D, <EOS>]에 대하여 출력열 [A->C->D-><EOS>]를 구하는 포인터 네트워크 모델이다. 인코더에서 입력열에 대한 인코딩($\overleftrightarrow{h}_s$)을 만들고, 이를 이용하여 디코더에서 입력열에 대응되는 위치(index)를 출력한다. 입력열에서 얼라인먼트 스코어가 가장 높은 위치를 굵은 화살표로 표시한다. 예를 들어, 디코더의 시작 입력인 A에 대한 출력 결과는 입력열에서 가장 높은 점수를 갖는 index 2 (즉, C)가 된다. 다음 디코딩은 C를 입력으로 하여 출력 결과 index 3 (즉, D)을 예측한다. D를 입력으로 한 디코더에서의 출력 결과는 index 4 (즉, <EOS>)가 되며 예측이 종료된다.

4. 멘션탐지

멘션은 같은 대상을 가리키는 단어들을 서로 참조해결하여 같은 개체로 군집화하기 위한 상호참조해결의 기본 입력 단위이다. 멘션은 보통 하나의 명사 또는 명사구로 구성되며, 한 문장 또는 하나의 구 안에 여러 멘션이 존재한다. 이와 같은 멘션들 중에 가장 바깥에 있는 멘션을 안은 멘션(outer mention), 그 안에 내포되어 있는 멘션을 안긴 멘션(inner mention)이라 하며, 규칙기반과 통계기반 시스템을 이용하여 멘션탐지를 수행한다. [표 1]은 멘션탐지에 대한 예를 보인다.

표 1. 멘션탐지 예

입력문장
난중일기(亂中日記)는 조선 중기의 무신(武臣) 이순신(李舜臣)이 임진왜란의 7년(1592 ~ 1598년) 동안 군중에서 썼다. 이것은 1962년 12월 20일 대한민국의 국보 제76호로 지정된 일기이다.
멘션탐지 결과
[난중일기(亂中日記)는] [[[조선 중기]의 무신(武臣)] 이순신(李舜臣)이] [[임진왜란의] 7년(1592 ~ 1598년) 동안] [군중에서] 썼다. [이것은] [[[[1962년 12월 20일] [대한민국의] 국보] 제76호로] 지정된 일기이다.]

[표 1]에서 모든 멘션은 어절단위로 명사와 명사구를 기반으로 추출된다. 여기서 멘션의 시작과 끝을 멘션 경계라 하며, 이 경계는 대괄호([])를 이용하여 표현한다. [표 1]에서 멘션의 경계는 안은 멘션의 경계(굵은 글씨 대괄호), 안긴 멘션의 경계(일반 글씨 대괄호) 등으로 나누어 구분한다. 예를 들어, "조선 중기의 무신(武臣) 이순신(李舜臣)"은 [조선 중기의 무신(武臣) 이순신(李舜臣)], [조선 중기의 무신(武臣)], [조선 중기]와 같은 멘션들로 정의되며, 중복된 멘션 중 길이가 가장 긴 [조선 중기의 무신(武臣) 이순신(李舜臣)]이 안은 멘션이고, 나머지 멘션들이 안긴 멘션이다.

안은 멘션과 안긴 멘션에 대한 출력 열을 BIO 태그로 표현하게 되면 [B-B-B, I-I-I, O-I-I, O-O-I]와 같이 타겟 클래스가 중복되는 문제가 발생하며, 중복되는 클래스를

모두 고려하여 학습을 수행하면 타겟 클래스 수가 많아져 학습이 어려워진다.

본 논문에서 제안하는 포인터 네트워크를 이용한 멘션 탐지의 출력 클래스는 BIO 태그가 아닌 입력 열에 대응되는 위치이다. 포인터 네트워크의 학습데이터 구조(4.1장 참고)는 멘션 경계의 시작 위치(begin index)와 마지막 위치(end index)로 구성되기 때문에 중첩되는 멘션(즉, 안은 멘션과 안긴 멘션의 관계)을 포함하여 학습을 수행할 수 있다.

4.1 멘션탐지 입력 방법

포인터 네트워크를 이용한 멘션탐지 입력 방법은 [표 2]와 같이 형태소 정보를 기준으로 하며, 모든 형태소 기준, 최적 형태소 기준 등 2가지 방법으로 정의한다.

표 2. 포인터 네트워크 학습에 대한 멘션탐지 입력 방법

입력 문장	
이곳에는 사람 얼굴 형상의 거대한 석상이 있는데 무엇일까?	
입력 문장에 대한 형태소 및 위치 정보	
이곳/NP:0 에/JKB:1 는/JX:2 사람/NNG:3 얼굴/NNG:4 형상/NNG:5 의/JKG:6 거대/NNG:7 하/XSA:8 ㄴ/ETM:9 석상/NNG:10 이/JKS:11 있/VA:12 는데/EC:13 무엇/NP:14 이/VCP:15 ㄹ까/EF:16 ?/SF:17 </S>:18	
입력 방법	**디코더 입력 구조**
모든 형태소 기준	[이곳/NP:0, 이곳/NP:0, </S>:18] [에/JKB:1, </S>:18] [는/JX:2, </S>:18] [사람/NNG:3, 사람/NNG:3, </S>:18] [얼굴/NNG:4, 사람/NNG:3, </S>:18] [형상/NNG:5, 사람/NNG:3, </S>:18] [의/JKG:6, </S>:18] [거대/NNG:7, </S>:18] [하/XSA:8, </S>:18] [ㄴ/ETM:9, </S>:18] [석상/NNG:10, 사람/NNG:3, </S>:18] [이/JKS:11, </S>:18] [있/VA:12, </S>:18] [는데/EC:13, </S>:18] [무엇/NP:14, 무엇/NP:14, </S>:18] [이/VCP:15, </S>:18] [ㄹ까/EF:16, </S>:18] [?/SF:17, </S>:18]
최적 형태소 기준	[이곳/NP:0, 이곳/NP:0, </S>:18] [사람/NNG:3, 사람/NNG:3, </S>:18] [얼굴/NNG:4, 사람/NNG:3, </S>:18] [형상/NNG:5, 사람/NNG:3, </S>:18] [거대/NNG:7, </S>:18] [ㄴ/ETM:9, </S>:18] [석상/NNG:10, 사람/NNG:3, </S>:18] [있/VA:12, </S>:18] [는데/EC:13, </S>:18] [무엇/NP:14, 무엇/NP:14, </S>:18] [?/SF:17, </S>:18]

본 논문에서 제안한 방법의 입력 기준은 형태소 정보이기 때문에 [표 2]와 같은 입력 문장을 형태소 정보(여기서는 각 형태소에 대하여 "원형/타입:위치_인덱스"로 표현한다)로 변환한다. 포인터 네트워크의 학습으로 사용할 데이터셋의 입력 방법은 모든 형태소를 대상으로 한 "모든 형태소 기준"과 실제 멘션탐지에서 사용될 형태소 정보만 대상으로 한 "최적 형태소 기준"등 2가지 방법으로 나뉜다. "입력 방법"에서 디코더의 입력은 [멘션의 중심어 경계 위치 -> 멘션의 시작 경계 위치 -> 종료 문자 위치]와 같이 구성된다. 멘션이 아닌 경우에는 [형태소 위치 -> 종료 문자 위치]와 같이 구성된다('</S>' 가 종료 문자임). 실제 디코딩 시에는 디코더의 입력 중에서 첫 번째 항목만 주어진다.

최적 형태소 기준은 실제 멘션의 경계로 적용되는 19개 형태소 정보만 추출하는 방법이며, 추출되는 형태소 타입은 다음과 같다. "EC, ETM, ETN, MAG, MAJ, MM, SF, SH, SL, SN, SS, SW, VA, VCN, VV, VX, XPN, XR, XSN".

5. 실험

실험에 사용된 데이터 셋은 뉴스 도메인 150 문서와 퀴즈 도메인 2,415 문서를 기반으로 하며 [표 3]과 같다.

표 3. 실험에 사용한 데이터셋

데이터 셋			
도메인	**학습(train)**	**개발(dev)**	**테스트(test)**
질의응답	1,935	240	240
뉴스	130	10	10

본 논문에서는 포인터 네트워크를 이용한 멘션탐지에 대하여 다음과 같이 실험을 하였다. 단어 표현은 10만 단어에 대한 세종 말뭉치를 NNLM (Neural Network Language Model)[9]으로 학습한 것을 사용하였다. 포인터 네트워크는 학습율(learning rate)을 0.05로 시작하여, 성능 개선이 없으면 5 에포크(epoch)마다 50%씩 감소하도록 정의하여 학습하였다. 히든 레이어와 어텐션 레이어에 대한 활성함수는 모두 tanh를 적용하였다.

멘션탐지의 성능 측정을 위하여, 출력 결과는 BIO 태그로 변환하여 측정하였으며, 성능 측정을 위한 척도(measure)는 F1 값을 사용하였다. 실험은 입력 방법에 따른 최적 성능과 스코어 함수 방법들(***concat***, ***concat2***)에 대한 비교실험, 하이퍼 파라미터 최적화, 멘션탐지 모델 비교 실험 등을 수행하였다.

상호참조해결은 앞서 수행한 멘션탐지의 결과를 기반으로 포인터 네트워크를 이용한 규칙기반 상호참조해결

[8] 성능을 측정하였으며, 성능 측정의 척도는 CoNLL F1 값[5]을 사용하였다(표 8 참고).

[표 4]는 포인터 네트워크의 입력방법에 따른 실험을 보인다([표 4, 5]의 히든 레이어 유닛 수는 [100, 50](즉, $[\overleftrightarrow{h}_s, h_t]$)을 기준으로 한다). 그 결과, 본 논문에서 제안한 "최적 형태소 기준" 입력 방법이 "모든 형태소 기준" 입력 방법에 비하여 약 0.05% 더 좋은 성능을 보였다.

표 4. 포인터 네트워크의 입력 방법에 따른 비교

입력방법	정확률	재현율	F1 (dev)
모든 형태소 기준	79.36	77.88	78.62
최적 형태소 기준	80.73	76.72	**78.67**

[표 5]는 alignment scoring 방법에 대한 비교 실험 결과이다. 개발셋에서 ***concat2***의 F1 값이 79.40%로 ***concat***에 비하여 약 0.73% 더 좋은 성능을 보였다.

표 5. alignment scoring 방법에 따른 성능 비교

모델	정확률	재현율	F1 (dev)
concat	80.73	76.72	78.67
concat2	**81.14**	**77.74**	**79.40**

[표 6]은 [표 5]에서 보다 좋은 성능을 보인 ***concat2***에 대하여 하이퍼 파라미터 최적화를 수행한 것이다. 개발셋에 대하여 히든 레이어 유닛 수가 [800, 400]일 때 F1 값이 80.21%로 가장 좋은 성능을 보였으며, 테스트셋에 대하여 80.75%의 F1 성능을 보였다.

표 6. *concat2*에 대한 포인터 네트워크 파라미터 최적화

Attention scoring method	$[\overleftrightarrow{h}_s, h_t]$의 차원 수	F1 (dev)	F1 (test)
concat2	[100, 50]	79.40	-
	[200, 100]	80.12	-
	[400, 200]	79.90	-
	[800, 400]	**80.21**	**80.75**
	[1600, 800]	79.65	-

[표 7]은 본 논문에서 제안한 방법과 규칙기반[5], Bi-LSTM-CRF 모델 기반[6] 등에 대한 멘션탐지 실험 결과를 보인다. 본 논문에서 제안한 방법이 긴 멘션에 대한 경우일 때는 F1 73.39%로 Bi-LSTM-CRF 보다 약 2.85% 낮은 성능을 보였지만, 규칙기반에 비하여 29.31% 높은 성능을 보였다. 또한 포인터 네트워크를 이용하는 경우에는 긴 멘션뿐만 아니라 모든 멘션에 대한 멘션탐지도 가능하다는 장점이 있다. 모든 멘션에 대하여 멘션탐지를 수행한 경우에는 규칙기반 보다 약 8.33% 높은 F1 80.75%의 성능을 보였다.

표 7. 멘션탐지 실험 결과

모델	긴 멘션 F1 (test)	모든 멘션 F1 (test)
규칙기반 [5]	44.08	72.42
Bi-LSTM-CRF [6]	**76.24**	-
포인터 네트워크	73.39	**80.75**

[표 8]은 한국어 멘션탐지와 영어 멘션탐지에 대한 성능비교이다. 영어 멘션탐지에는 ACE 2005 데이터셋을 사용하고, BILOU (B: beginning, I: inside, L: last, O: outside, U: unit) 태그를 사용한 Joint System [10, 11]과 BIDIRECT [12]의 성능을 보인다. [10]은 파이프라인(pipe-line) 기반으로 엔티티의 멘션과 관계(relation)를 추출하는 방법이고, [11]은 정보 네트워크(information networks)를 이용하여 엔티티의 멘션, 관계, 이벤트(event)를 추출하는 방법이며, [12]는 Bi-directional RNN을 이용하여 멘션을 추출하는 방법이다. [표 8]에 따라 영어 멘션탐지가 본 논문의 방법보다 더 높은 성능을 보이지만, [10, 11]은 사람이 직접 자질 추출 및 조합을 수행하였으며, [10, 11, 12] 모두 BILOU 태그를 이용하기 때문에 중첩되는 멘션에 대한 처리가 불가능하다.

표 8. 한국어 멘션탐지와 영어 멘션탐지의 성능 비교

멘션탐지 모델	정확률	재현율	F1 (test)
Joint System [10]	85.20	76.90	80.80
Joint System [11]	85.10	77.30	81.00
BIDIRECT [12]	83.70	81.80	82.70
포인터 네트워크 기반 멘션	81.82	79.71	80.75

[표 9]는 본 논문에서 제안한 포인터 네트워크를 이용한 멘션탐지 기반 상호참조해결 결과에 따른 성능을 보인다. 상호참조해결[5, 8]은 규칙기반(질문문서의 대명사 상호참조해결에 포인터 네트워크 적용) 성능을 측정하였으며, 멘션 경계 결과(Mention Boundary)를 보인다. 멘션 경계 결과는 상호참조 결과에서 탐지된 멘션의 경계를 유지하여 성능을 측정한 것이다. 실험 결과, 포인터 네트워크를 이용한 멘션탐지 기반 상호참조해결의 경우에는 규칙기반 보다 약 2.29%향상된 CoNLL F1 52.69%의 성능을 보였다. 이에 따라, 본 논문에서 제안한 방법이 상호참조해결 성능 향상에 도움이 되는 것을 알 수 있다.

표 9. 포인터 네트워크를 이용한 멘션탐지 기반 상호참조해결 실험 결과

멘션탐지 모델	CoNLL F1
규칙기반 멘션[5, 8]	50.40
포인터 네트워크 기반 멘션	**52.69**

[표 10]은 규칙기반 멘션탐지를 이용한 상호참조해결

와 본 논문에서 제안한 방법을 이용한 상호참조해결에 관한 결과를 보인다. 각 멘션은 대괄호([])를 이용하여 구분되며, 대괄호에 붙은 아래첨자는 멘션의 인덱스를 나타내고, 위첨자는 엔티티(entity)의 인덱스를 나타낸다. 엔티티는 서로 참조해결 되는 단어들의 군집(cluster)이다.

표 10. 상호참조해결 비교

규칙기반 멘션탐지를 이용한 상호참조해결
[이것은]$^{0}_{0}$ [고려 말기부터]$^{1}_{1}$ [여러 [사람의]$^{2}_{2}$ 시조를]$^{3}_{3}$ 모아 [1728년에]$^{4}_{4}$ 엮은 [고시조집이다.]$^{3}_{5}$
포인터 네트워크를 이용한 멘션탐지 기반 상호참조해결
[이것은]$^{0}_{0}$ [[고려 말기부터]$^{1}_{1}$ [[여러 사람의]$^{2}_{2}$ 시조를]$^{3}_{3}$ 모아 [1728년에]$^{4}_{4}$ 엮은 고시조집이다.]$^{0}_{5}$

[표 10]에서 규칙기반 멘션탐지를 이용한 상호참조해결의 경우, [여러 [사람의]$^{2}_{2}$ 시조를]$^{3}_{3}$ 와 같이 멘션의 크로스가 발생하거나, [고시조집이다.]$^{3}_{5}$ 와 같이 수식어 정보를 잃어버린 멘션이 되는 경우가 있다. 이와 같은 경우에는 수식어 부재로 인하여 잘못된 상호참조해결([사람의 시조를]$^{3}_{3}$과 [고시조집이다.]$^{3}_{5}$가 서로 참조 됨)이 발생한다. 반면에, 포인터 네트워크를 이용한 멘션탐지 기반 상호참조해결의 경우에는 모든 멘션의 경계가 올바르게 정의될 수 있기 때문에 앞서 발생한 문제들이 해결되어, 올바른 상호참조해결([이것은]$^{0}_{0}$과 [고려 ~ 고시조집이다.]$^{0}_{5}$이 서로 참조 됨)이 수행된다.

6. 결론

본 논문에서는 포인터 네트워크를 이용하여 중첩된 멘션을 포함한 멘션탐지를 수행하고, 상호참조해결 시스템에 적용하였으며, 포인터 네트워크의 입력 방법 2가지(모든 형태소 기준, 최적 형태소 기준)와 스코어 함수 2가지(***concat***, ***concat2***)에 대한 실험을 수행하였다. 실험 결과, 포인터 네트워크를 이용한 멘션탐지가 최적 형태소 입력 기준, concat2, [800, 400]에서 F1 80.75%로 가장 좋은 성능을 보였으며, 이에 기반한 상호참조해결이 CoNLL F1 52.69%로 기존의 규칙기반 멘션탐지를 이용한 상호참조해결보다 약 2.29% 더 좋은 성능을 보였다. 이에 따라, 본 논문에서 제안한 방법이 기존의 규칙기반 멘션탐지에 비하여 더 좋은 성능을 보임을 알 수 있으며, 상호참조해결에 도움이 되는 것을 알 수 있다.

향후 연구로는 상호참조해결 및 멘션탐지에 대한 학습 데이터를 더 구축할 것이며, 포인터 네트워크 모델을 hierarchical RNN 등과 같은 모델로 개선하여 상호참조해결 및 멘션탐지에 적용할 예정이다. 또한 포인터 네트워크를 개체명 인식 등과 같은 다른 경계탐지 문제 및 포인팅 개념을 적용할 수 있는 자연어처리 문제(문서 요약, 의존구문트리, 의미역 결정 등)에 적용할 예정이다.

감사의 글

이 논문은 2016년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임. (No.R0101-16-0062, (엑소브레인-1세부) 휴먼 지식증강 서비스를 위한 지능진화형 WiseQA 플랫폼 기술 개발)

참고문헌

[1] K. Cho, et al. Learning phrase representation using RNN encoder-decoder for statistical machine translation. *Proc. of EMNLP' 14*, 2014.

[2] D. Bahdanau, et al. Neural machine translation by jointly learning to align and translate. *Proc. of ICLR' 15*, arXiv:1409.0473, 2015.

[3] C. Gulcehre, et al. Pointing the Unknown Words. *arXiv:1603.08148* (2016).

[4] R. Nallapati. et al. Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond. *arXiv:1602.06023v2* (2016).

[5] 박천음, 최경호, 이창기. Multi-pass Sieve를 이용한 한국어 상호참조해결. *정보과학회논문지 41.11*, pp. 992-1005, 2014.

[6] 박천음, 이창기. Bidirectional LSTM-CRF 모델을 이용한 멘션탐지. *제 27회 한글 및 한국어 정보처리 학술대회 논문집*, pp. 224-227, 2015.

[7] 박천음, 이창기, 딥 러닝을 이용한 상호참조해결 멘션탐지. *제9회 한국정보과학회 한국빅데이터학회 공동학술 심포지엄*, 2015.

[8] 박천음, 이창기. 포인터 네트워크를 이용한 대명사 상호참조해결. *2016년 한국컴퓨터종합학술대회 논문집*, pp. 699-701, 2016.

[9] 이창기, 김준석, 김정희. 딥 러닝을 이용한 한국어 의존 구문 분석. *제 26회 한글 및 한국어 정보처리 학술대회*, pp. 87-91, 2014.

[10] Q. Li, et al. Incremental Joint Extraction of Entity Mentions and Relations. *Proc. of ACL' 14*, 2014.

[11] Q. Li, et al. Constructing information networks using one single model. *Proc. of EMNLP' 14*, 2014.

[12] TH. Nguyen, et al. Toward Mention Detection Robustness with Recurrent Neural Networks. *arXive preprint arXiv:1602.07749*, 2016.