가상 엔터티 설명문 및 엔터티 정렬에 기반한 엔터티 링킹 전이학습

최형준 [O1] 나승훈 [1] 김현호 [2] 김선훈 [2] 강인호 [2]
[1] 전북대학교 [2] 네이버
modera20172gmail.com, nash@jbnu.ac.kr,
{kim.hh, seonhoon.kim, once.ihkang}@navercorp.com

Transfer learning of Entity linking based on Pseudo Entity Description and Entity Alignment

Heyon-Jun Choi[O1] Seung-Hoon Na[1] Hyun-Ho Kim[2] Seon-Hoon Kim[2] Inho Kang[2]
[1]Jeonbuk National University [2]Naver

요 약

엔터티 링킹을 위해서는 엔터티 링킹을 수행 할 후보 엔터티의 정보를 얻어내는 것이 필요하다. 하지만, 엔터티 정보를 획득하기 어려운 경우, 엔터티 링킹을 수행 할 수 없다. 이 논문에서는 이를 해결하기 위해 데이터셋으로부터 엔터티의 가상 엔터티 설명문을 작성하고, 이를 통해 엔터티 링킹을 수행함으로써 엔터티 정보가 없는 환경에서도 2.58%p밖에 성능 하락이 일어나지 않음을 보인다.

주제어: RoBERTa, Entity Linking, Transfer Learning

1. 서론

엔터티 링킹(Entity Linking)이란 주어진 문서 상에서 엔터티가 내포된 부분에 어떤 엔터티가 연결되어야 하는지를 찾아내는 작업을 의미한다. 따라서, 엔터티 링킹을 위해서는 다음 두 가지가 전제되어야 한다: 첫번째는 문서 내에서 엔터티에 해당하는 부분을 찾아내는 개체명 인식이고, 두 번째는 엔터티 링킹을 수행 할 후보 엔터티의 정보를 추려내는 것이다. 일반적으로 엔터티의 정보는 위키백과 등의 대량의 문서 집합이나 지식베이스를 통해 얻을 수 있다. 하지만 이런 방법은 엔터티가 지식베이스에 포함이 되있는 경우를 전제로 하기 때문에, 엔터티가 포함이 안된 경우 활용을 할 수 없다.

이 논문에서는 이렇게 정보를 얻을 수 없는 엔터티에 대응하기 위해 *가상 엔터티 설명문(Pseudo Entity Description)*을 제시 할 것이다. 가상 엔터티 설명문은 엔터티 링킹을 위해 필요한 엔터티에 대한 설명 대신, 데이터셋 내의 해당 엔터티가 등장한 데이터를 엔터티 설명 대신 사용함으로써 엔터티에 대한 정보를 획득 할 수 없는 상황에 대처 할 수 있다. 이러한 환경을 실험하기 위해 이전 연구[1,2]에서 사용한 한국어 위키백과 데이터셋과, 학습된 모델을 사용하였다. 이때, 엔터티 설명에 해당하는 부분을 일부 제외하고, 대신 3.1절의 방법을 통해 학습 데이터셋에서 엔터티에 대한 가상 엔터티 설명문을 생성하여 대신 사용하였다.

2. 관련 연구

엔터티 링킹은 꾸준히 연구되어온 주제이다. [3]에서는 GCN을 사용하여 기존의 방법에 비해 엔터티 성능을 향상시켰다. [2]에서는 기존의 엔터티 링킹 방법처럼 개체명 인식을 전제로 하지 않고, 엔터티 링킹시 NIL 엔터티를 두어 개체명 인식과 엔터티 링킹을 동시에 수행하였다. 이 논문에서는 [2]의 방법을 사용하여 엔터티 링킹을 구현하고, 학습된 모델을 가상 엔터티 설명문으로 전이학습을 시행하였다.

[2]에서는 엔터티 설명을 RoBERTa 출력을 통해 사용했지만, RELIC[4]은 엔터티 임베딩을 직접 학습시키는 방법을 제시했다. 이는 대량의 말뭉치를 통해 엔터티 임베딩 자체를 사전학습 시키는 방법을 제시한다. 이렇게 엔터티 임베딩을 학습시키는 다른 방법으로 EaE[5]가 있다.

3. Nil-aware 엔터티 링킹

3.1 데이터 양식

엔터티 링킹을 위해 한국어 위키백과를 사용하였다. 위키백과 문서에서 하이퍼링크가 걸려 있는 부분을 *멘션(Mention)*으로, 해당 하이퍼링크가 가리키는 문서를 *엔터티(Entity)*로 사용하였다. 이때, 모든 엔터티와 멘션 쌍을 통해 각 멘션이 나타낼 수 있는 엔터티-멘션 집합 $E(m)$을 구축한다. 이 집합은 멘션 m에 대해 전체 데이터셋에서 멘션이 m 인 경우 등장한 모든 엔터티 $\{e_1, e_2, \ldots\}$를 가리킨다.

NIL 엔터티를 추가하기 위해 토크나이징된 각 데이터

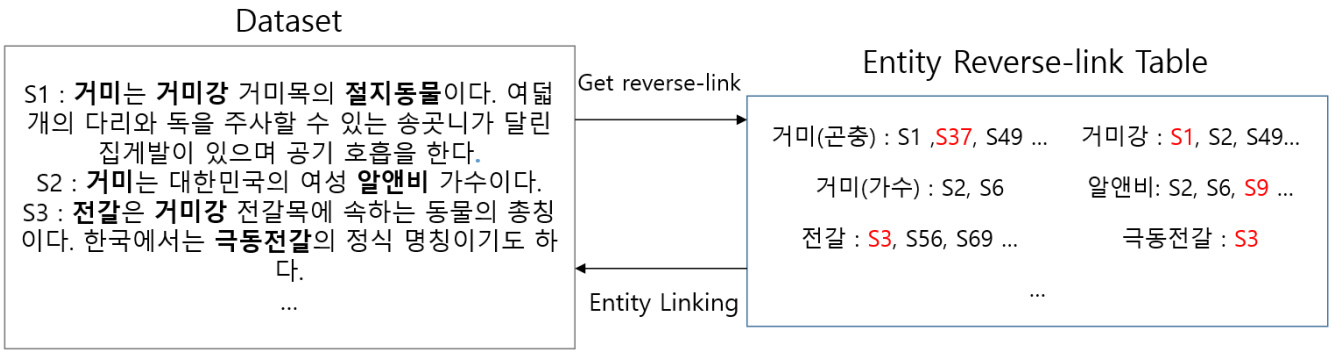

그림 1 가상 엔터티 설명문 생성 예시

에 대해 n-gram 매칭을 통해 문장 의 각 n-gram 스팬이 엔터티-멘션 집합에 포함 돼 있는 지를 판단, 포함이 된 경우, 멘션으로 취급한다. 이때, 실제로 엔터티가 링킹 되어야 하는 경우를 non-Nil, 엔터티가 링킹되지 않은 경우를 Nil로 취급한다. 작업의 효율성을 위해 n-gram은 unigram부터 10-gram까지 비교했다. 각 멘션 스팬은 시작지점 x_{start}과 끝점 x_{end}, 엔터티 e로 구성되며, Nil 멘션인 경우 Nil 엔터티 $e_{\emptyset}$가 대신 추가된다.

각 멘션 스팬은 엔터티 링킹 수행을 위해 엔터티 후보가 추가된다. 엔터티 후보는 E(m)를 통해 획득하고, 항상 $e_{\emptyset}$를 추가하여 해당 멘션의 Nil 여부를 구별한다. 학습 시에는 최대 10개의 엔터티 후보가 존재하게 하였고, 이를 초과하는 경우, 정답 엔터티가 포함되는 선에서 랜덤으로 추출했다. 부족한 경우 랜덤으로 엔터티를 추가하여 $e_{\emptyset}$포함 총 11개의 엔터티로 구성되게 하였다

따라서 각 데이터는 다음과 같이 구성된다.

- 토크나이징 된 문장: $X = [x_1, x_2, \dots, x_i]$
- 멘션 스팬: $S = [(x_{start}, x_{end}, e, [e_{\emptyset}, e_1, e_2, \dots]), \dots]$

3.2 엔터티 임베딩

*엔터티 임베딩*은 각 엔터티에 대응하는 표상을 저장해 둔 집합이다. 각 엔터티에 대한 표상은 위키백과의 각 엔터티 문서를 토크나이징 한 후 문서 맨 처음에 <s> 토큰을 추가 한 후, RoBERTa를 통해 인코딩 한 후 <s>토큰에 해당하는 출력을 저장하여 구현한다. Emb(e)는 엔터티 e에 대한 표상 X'_e을 나타내는 엔터티 임베딩이다.

$$X_e = [< s >, e_1, \dots, e_i, </s >]$$
$$X'_e = \text{RoBERTa}(X_e)[0]$$
$$\text{Emb}(e) = X'_e$$

3.3 엔터티 링킹 모델

엔터티 링킹 모델은 RoBERTa을 통해 입력 문장 X를 인코딩 한 후, 문장 내의 각 멘션 스팬 S의 시작 토큰 x_{start}과 x_{end}을 concatenate한 후, 이를 엔터티 정보와 Biaffine[6] 연산을 통해 각 엔터티의 점수를 계산한다. 아래 식 중 (1)번 식이 Biaffine 연산이다. 여기서 U와 W_d, W_e는 가중치이고 **b**는 바이어스이다. 엔터티 정보는 위키백과의 각 엔터티 문서를 RoBERTa를 통해 인코딩 한 후, <s>토큰이 존재하는 가장 첫 번째 토큰의 출력을 사용하였다. 엔터티 정보는 학습 전에 미리 RoBERTa를 통해 인코딩 한 후, 학습 중에는 값을 고정시켜 학습되지 않도록 한 후 사용하였다.

$$X' = \text{RoBERTa}(X) = [x'_1, \dots, x'_i]$$
$$\text{Score}(e) = \text{Biaffine}(\text{concat}(x'_{\text{start}}, x'_{\text{end}}), \text{Emb}(e))$$
$$\text{Biaffine}(\mathbf{e}, \mathbf{d}) = \mathbf{d} \cdot U \cdot \mathbf{e}^T + W_d \cdot \mathbf{d} + W_e \cdot \mathbf{e} + \mathbf{b} \quad (1)$$

3.3 가상 엔터티 설명문

3.2 절과 같이 위키백과의 문서나 다른 지식 베이스를 통해 엔터티에 대한 정보를 획득할 수 없는 경우를 상정하기 위해 데이터셋으로부터 가상의 문서를 만들어낸다. 3.1절에서 E(m)를 구성하는 과정에서 각각의 문서와 그 문서에 등장한 엔터티에 대한 역링크 테이블을 작성, 각각의 엔터티가 어떤 문서에서 등장했는지를 기록한다. 이후 가상 엔터티 설명문을 만들 엔터티에 대해 해당 엔터티가 등장 한 문서를 임의로 선택 그 문서를 해당 엔터티의 가상 엔터티 설명문으로 취급한다. 그림1은 이 과정에 대한 예시이다. 이렇게 만들어진 가상 엔터티 설명문을 엔터티 정보로 사용하여 3.2절의 과정을 통해 엔터티 링킹을 수행한다.

3.4 엔터티 얼라인먼트

*엔터티 얼라인먼트*는 가상 엔터티 설명문과 엔터티 id를 연결해주는 작업을 의미한다. 이는 등장한 적 없는 멘션에 대해 기존 엔터티 정보를 매핑하는 식으로 동작한다. 이를 위해 미리 학습시킨 [2]의 엔터티 링킹 모델을 활용했다. 우선 학습된 엔터티 링킹 모델을 통해 엔터티 링킹을 시행한다. 이때 E(m)를 통해 멘션 후보를 추출하는 것이 아니라, 등장한 적 없는 멘션을 후보로 입력한다. 모델의 출력에서 가장 높은 확률을 가진 엔터

티를 등장한 적 없는 멘션에 링킹되는 엔터티로 간주, E(m)을 확장한다. 만약, 이 과정에서 $e_{\emptyset}$을 출력한 경우 해당 멘션과 가상 엔터티 설명문 e_p를 E(m)에 추가한다.

3.5 전이학습

3.4절을 통해 확장된 E(m)를 이용해서 데이터셋을 재구축, 학습된 엔터티 링킹 모델을 통해 전이학습을 시행한다. 3.4절에서 사용한 학습된 모델을 불러와서 학습을 진행하였고, 일부 엔터티를 누락시키는 식으로 엔터티 설명문의 부재 환경을 시험했다. 누락시키는 비중은 4.1절과 같이 여러 비중으로 실험을 진행했다.

4. 실험

4.1 데이터 구성

실험을 위해 한국어 위키백과를 통해 [2]과 같이 데이터셋을 구성하였다. 총 6만의 문서를 사용하였고, 이중 3만개를 train, 2만개를 test, 1만개를 dev 셋으로 사용하였다. 또한, 2217개의 데이터로 구성된 네이버의 엔티티 링킹 데이터셋을 통해 실험을 진행했다. 이 데이터셋은 뉴스 기사등에서 수작업으로 멘션 부분과 엔티티 id를 표시해둔 것으로, 위키백과의 하이퍼링크와 달리 가게 이름, 인명 등의 저명성이 떨어지는 엔터티가 포함되어 있다. 또한, 이에 대한 엔터티 설명이 전혀 존재하지 않는 엔티티가 전체 멘션의 91.24%를 차지하기 때문에, 이러한 경우 가상 엔터티 설명문에 의존하여 엔터티 링킹을 시행하였다. 이중 70%를 train, 20%를 test, 10%를 dev셋으로 사용하였다. 각 문서는 토큰 단위로 세서 최대 길이가 50이 되도록 했고, 이를 초과 하는 경우, 2개 이상의 문장으로 나뉘도록 했다. 이때, 나눠지는 문장들은 25개 토큰이 서로 겹쳐지도록 했다. 문장단위로 나눴을 때 train셋은 14,128개, test셋은 7088개, dev 셋은 2484개의 문장으로 구성된다.

각 입력 문장은 한국어 형태소 분석기를 통해 형태소 분석이 적용되었고, 여기에 추가로 BPE[7]를 통해 한번 더 토크나이징을 했다. 엔터티 링킹 모델은 [2]와 동일한 구조를 사용하되, RoBERTa[8]를 통해 인코딩된 엔터티 설명 대신 가상 엔터티 설명문을 RoBERTa를 통해 인코딩한 것을 사용하였다. RoBERTa의 구성은 BERT-base[9]와 동일하다.

가상 엔터티 설명문은 해당 엔터티가 등장한 데이터셋 상의 문장을 임의로 하나를 선택하여 RoBERTa를 통해 인코딩, <s> 토큰이 존재하는 가장 첫번째 토큰의 출력을 사용했다. 비교를 위해 가상 엔터티 설명문으로 변환하는 비중을 10%~50%, 100%로 변환하였고, 변환되는 엔터티는 임의로 선택하였다.

엔터티 링킹을 실험 하기 전에 같은 데이터셋을 통해 미리 학습된 엔터티 링킹 모델의 파라미터를 불러와서 사용하였다. 이는 Baseline의 파라미터와 동일하다. 하지만, 엔터티 임베딩은 제거하여 가상 엔터티 설명문을 포함 한 새 엔터티 임베딩으로 치환하여 사용했다.

4.2 실험결과

표 1 가상 엔터티 설명문 변환 비중 별 엔터티 링킹 성능

변환 비중	정밀도	재현율	F1	Nil 탐지 F1
Baseline[2] (0%)	85.96%	85.78%	85.87%	89.05%
10%	85.53%	84.69%	85.11%	88.68%
20%	84.95%	84.10%	84.52%	88.48%
30%	84.56%	83.59%	84.07%	88.31%
40%	84.39%	82.94%	83.66%	88.27%
50%	84.33%	82.31%	83.31%	88.21%
100%	84.43%	82.19%	83.29%	88.18%

표1은 각 가상 엔터티 설명문 변환 비중 별 엔터티 링킹 성능이다. Nil 탐지 F1는 해당 멘션이 Nil멘션인지 아닌지를 판별하는 F1 점수를 의미하며, 모델이 예측 한 것이 Nil 엔터티인지, Nil 엔터티가 아닌 다른 엔터티인지를 통해 측정하였다. 가상 엔터티 설명문은 온전한 엔터티 정보에 비해 해당 엔터티의 정보를 잘 나타내지 않기 때문에 변환 비중이 증가할수록 성능이 낮아지지만, 50%을 변환 했음에도 F1 성능은 2.58%p밖에 하락하지 않음을 볼 수 있고, 재현율을 제외한 다른 성능 지표는 크게 하락하지 않음을 확인할 수 있다.

표 2 전이학습 실험 결과 - 네이버 데이터셋

데이터셋	정밀도	재현율	F1	Nil 탐지 F1
dev	49.92%	66.49%	57.02%	65.52%
test	69.42%	85.97%	76.81%	81.52%

표2는 네이버의 엔터티 링킹 데이터셋을 사용하여 전이학습을 시행한 결과이다. 데이터 편차가 매우 크고 데이터셋이 작기 때문에 dev셋과 test셋의 성능 편차가 상당히 크다. 하지만, 이러한 환경에서도 엔터티 링킹이 어느정도 수행 가능함을 볼 수 있다.

5. 결론

본 논문에서는 데이터셋으로부터 가상 엔터티 설명문을 구성하는 방법을 제시하고, 이를 통해 엔터티 링킹 성능을 측정함으로써 이 방법을 통해 엔터티 정보를 획득할 수 없는 환경에서도 엔터티 링킹 성능이 2.58%p밖에 하락하지 않음을 보였다.

본 논문에서는 전이학습 시 단 하나의 페이지로만 매핑을 했다. 하지만, 엔터티 링킹 모델의 출력이 각 엔터티에 대한 확률값임을 이용해서 기존 엔터티 임베딩과 새로 만들어진 엔터티 임베딩 중 더 적합한 것을 선택하도록 할 수 있다. 이를테면, 기존의 임베딩이 E, 가상 엔터티 설명문을 통해 구축된 새 임베딩이 E′라고 하면,

엔터티 얼라인먼트를 통해 기존의 임베딩인 E를 사용할지에 대한 여부를 계산한다. 만약, 기존의 모델을 통한 출력이 기존의 엔터티e를 가리킨다면, 그 엔터티를 모델의 출력으로 사용하고, $e_{\emptyset}$를 가리킨다면 E′를 통해 엔터티 링킹을 수행하여 확장된 엔터티 임베딩을 사용한다. E′를 사용하여 엔터티 링킹을 수행한 결과가 $e_{\emptyset}$인 경우, 해당 멘션은 Nil 멘션이 된다.

6. 참고문헌

[1] 최형준, 나승훈, 김현호, 김선훈, 강인호, "NIL 탐지형 멘션 추출 및 개체 링킹", 한국 정보과학회 학술발표논문집, 2020.7

[2] 홍승연, 나승훈, 김현호, 김선훈, 강인호, "멘션 임베딩을 이용한 NIL 멘션 탐지와 개체 연결의 통합 모델", 한국 정보과학회 학술발표논문집, 2020.7

[3] 이영훈, 나승훈, "Graph Convolutional Network 기반 집합적 개체 연결", 제 31회 한글 및 한국어 정보처리 학술대회, 2019.10

[4] Jeffrey Ling, et al. “Learning Cross-Context Entity Representations from Text”, ICLR, 2020

[5] Févry, Thibault, et al. "Entities as experts: Sparse memory access with entity supervision." arXiv preprint arXiv preprint arXiv:2001.03765, 2020

[6] Dozat, Timothy, and Christopher D. Manning. "Deep biaffine attention for neural dependency parsing." arXiv preprint arXiv:1611.01734, 2016

[7] Rico Sennrich, Barry Haddow, et al, “Neural Machine Translation of Rare Words with Subword Units”, 2015.08

[8] Liu, Yinhan, et al. "Roberta: A robustly optimized bert pretraining approach." arXiv preprint arXiv:1907.11692, 2019

[9] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805, 2018