질의를 위한 효율적인 Bi-encoder기반 End-to-End 한국어 개체 연결

이성민, 나승훈

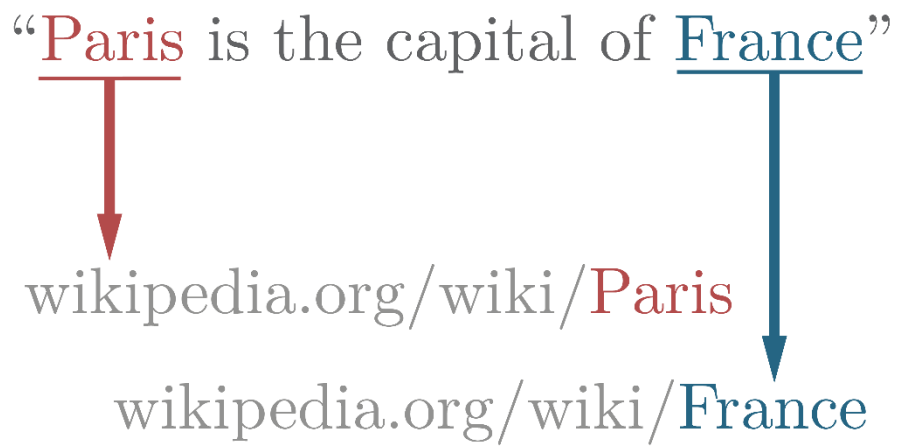
인지컴퓨팅 연구실

전북대학교

목차

- 배경
- 관련 연구
- 모델 구조
- 실험, 결과
- 결론

개체 연결

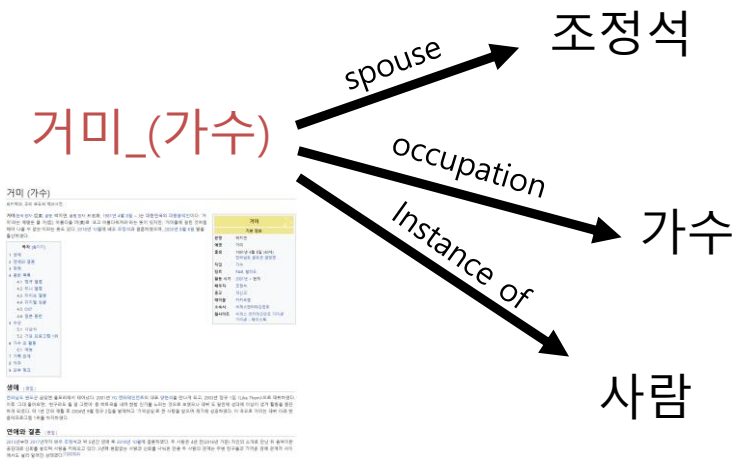


개체 연결은 문서 내에 등장하는
개체 멘션을 찾아내고 이를
지식베이스 상의 개체로 연결하는 작업이다.

개체 연결 for Question

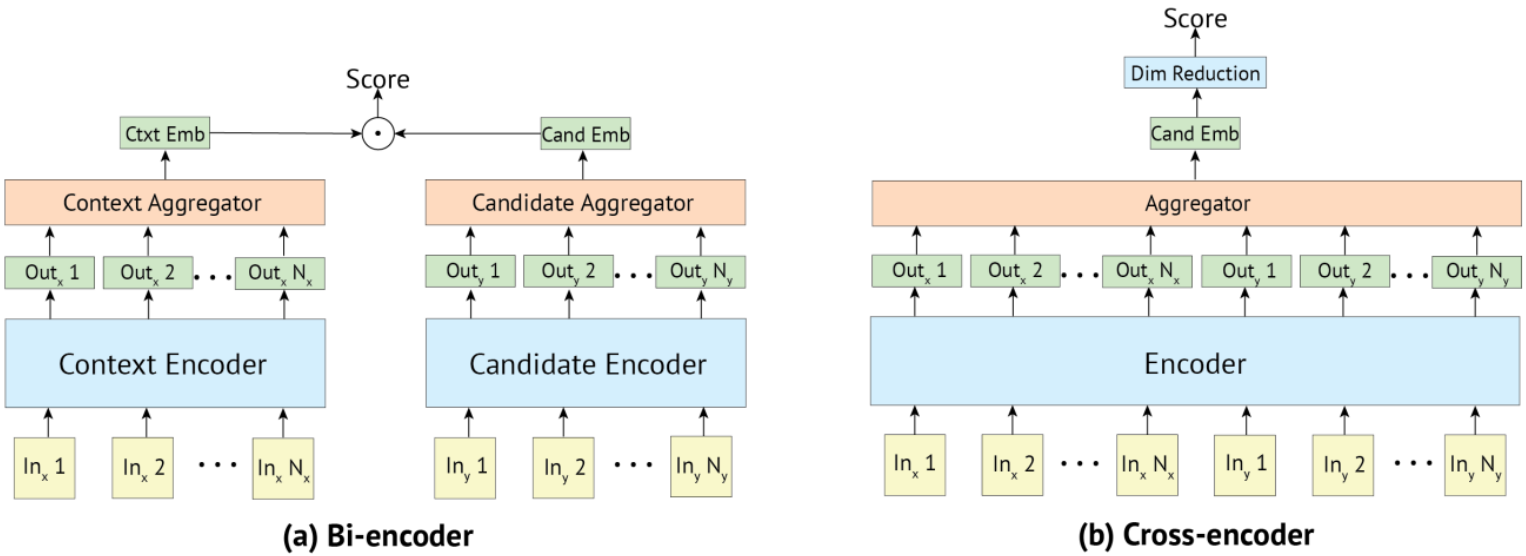
가수 거미는 누구야?

wikipedia.org/wiki/거미_(가수)



단문인 질의에 대해 정확한 개체연결을 할 수 있다면
예측한 엔티티의 타입, 아티클, 관계 등을
질의응답을 위한 자원으로 활용할 수 있다.

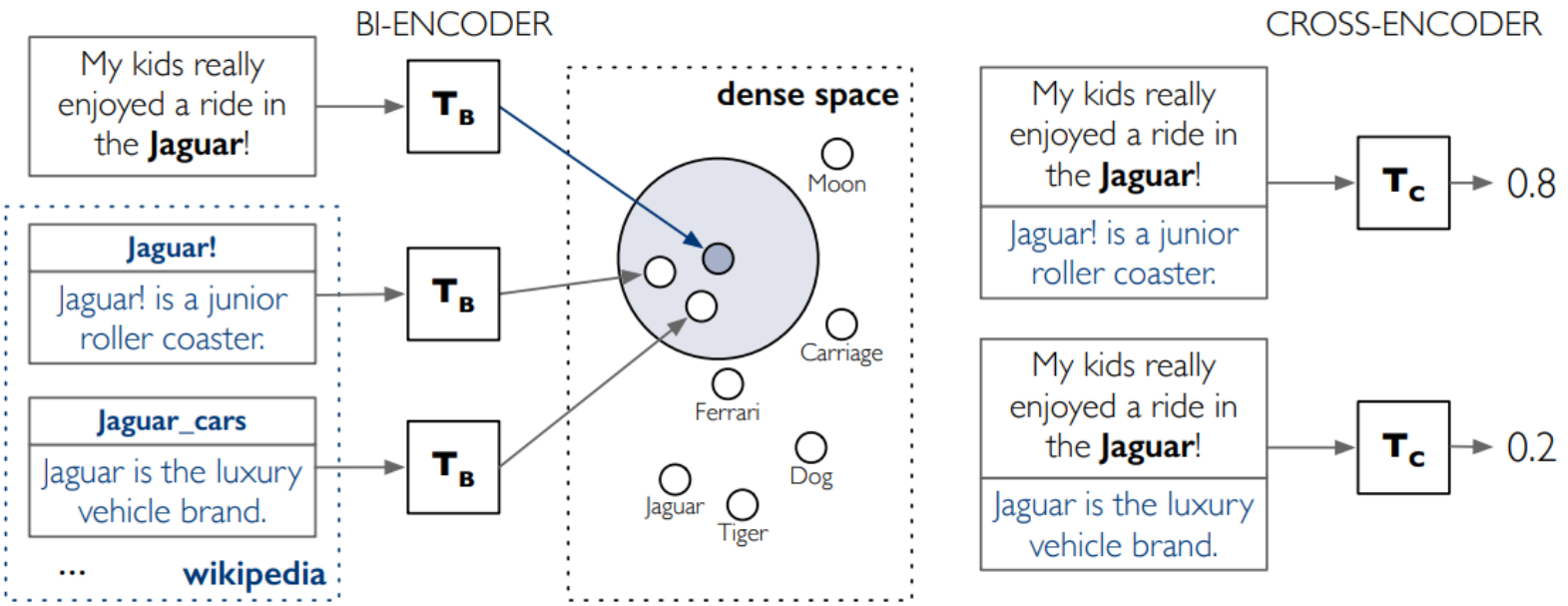
Poly-encoders: Transformer Architectures and Pre-training Strategies for Fast and Accurate Multi-sentence Scoring (S. Humeau el al. 2020)



(a) Bi-encoder (b) Cross-encoder

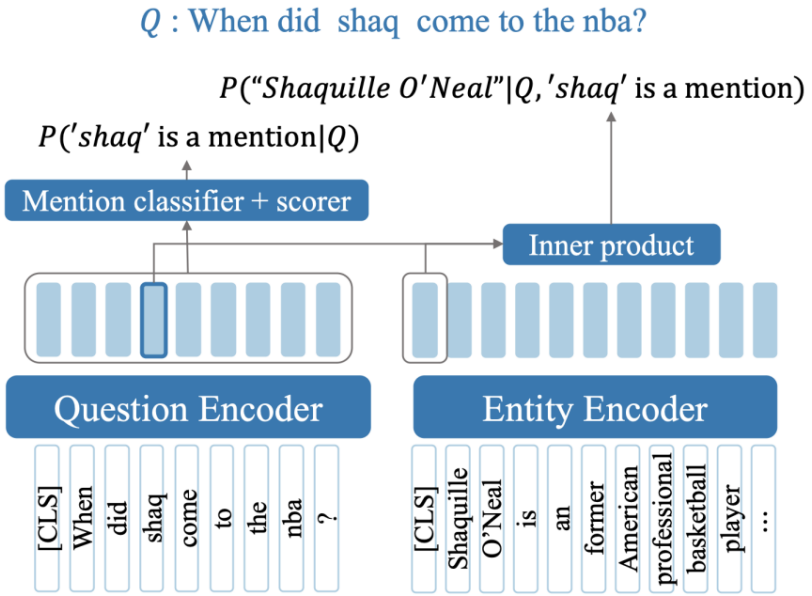
- Bi-encoder, Cross-encoder
  - Bi-encoder는 인코더를 두개로 분리하게 되면서 context, candidate 간의 Cross-attention이 없어 성능은 낮아지지만, 빠른 inner-product 연산을 통해 많은 Candidate에 대해 빠른 연산이 수행가능한 장점이 있다.
  - Cross-encoder는 반대로 Cross-attention으로 성능은 높지만 모든 Candidate에 대한 연산을 위해서는 많은 연산비용과 시간이 든다는 단점이 있다.

Scalable Zero-shot Entity Linking with Dense Entity Retrieval (L. Wu et al. 2020)



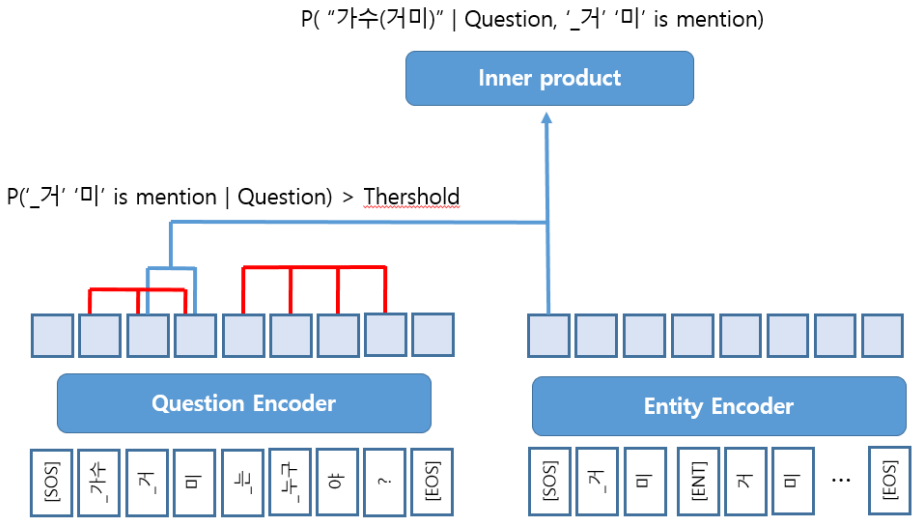
- 모델의 특징
  - Bi-encoder를 이용한 Top-k Candidate를 구하고 그것들을 Cross-encode를 통해 Score를 구함으로써 Bi-encoder의 빠른 속도와 낮은 비용, Cross-encoder의 Cross-attention을 통한 높은 정확도를 적절히 이용한 것을 볼 수 있다.
  - 하지만, Cross-encode를 사용하긴 했으므로 Encoder의 여러 번의 추론은 불가피하다.

Efficient one-pass end-to-end entity linking for questions (Belinda Z. et al. 2020)



- 모델의 특징
  - Bi-encoder를 이용하여 후보 엔티티들을 사전에 인코딩해둠으로써 Question Encoder의 한번의 추론으로 모든 후보들에 대한 평가가 가능하다.
  - 멘션탐지와 중의성해결을 한번에 해결하는 End-to-End모델로써, 여러 멘션과 후보 엔티티를 유동적으로 비교할 수 있기 때문에 앞선 cross-encoder를 이용한 모델에 많이 뒤쳐지지 않는 성능을 보임.

모델 구조 (Bi-Encoder기반)



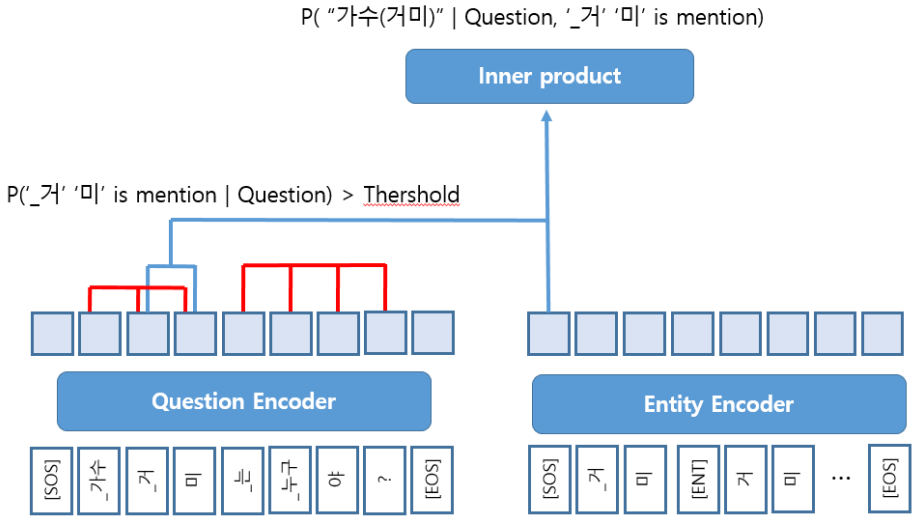
- 멘션 탐지

$$s_{start}(i) = \mathbf{w}_{start}^{\top}\mathbf{q}_i, \quad s_{end}(j) = \mathbf{w}_{end}^{\top}\mathbf{q}_j$$

$$s_{mention}(t) = \mathbf{w}_{mention}^{\top}\mathbf{q}_t$$

$$p([i,j]) = \sigma(s_{start}(i) + s_{end}(j) + \sum_{t=i}^{j} s_{mention}(t))$$

모델 구조 (Bi-Encoder기반)



- 중의성 해결

$$\mathbf{y}_{i,j} = \frac{1}{(j-i+1)} \sum_{t=i}^{j} \mathbf{q}_t$$

$$s(e,[i,j]) = \mathbf{x}_e^{\top} y_{i,j}$$

$$p(e|[i,j]) = \frac{\exp(s(e,[i,j]))}{\sum_{e' \in E} \exp(s(e',[i,j]))}$$

모델 구조 (Bi-Encoder기반)

- 학습

$$Loss_{MD} = -\frac{1}{N} \sum_{\substack{1 \le i \le j \le \\ min(i+L-1,n)}} (y_{[i,j]} \log p([i,j]) + 1(-y_{[i,j]}) \log(1 - p([i,j])))$$

$$Loss_{ED} = -\log p(e_g|[i,j])$$

실험, 결과

- 데이터셋

표 1: 위키피디아 개체 연결 데이터 셋 통계 (최대문서길이=300)

텍스트 개수	멘션 개수	엔티티 개수
485842	3040058	380588

표 2: KT 질의 데이터 셋 통계

Train	Dev	Test
3708	238	425

[KT 질의데이터셋 sample]

ex) Text: 이탈리아의 대통령이 누구일까요?
Seed_Entity: 이탈리아
Wikidata: Q38

실험, 결과

- 학습 과정
  1. In-batch negative mining을 적용하여 모델을 2 Epoch 학습
  2. Entity Encoder의 파라미터들을 고정시킨 뒤, 모든 후보엔티티들의 임베딩을 계산
  3. Entity Encoder는 사용하지 않고 계산해놓은 후보엔티티의 임베딩을 이용해 Hard negative mining을 적용하여 학습을 빠르게 진행
     (KT질의데이터셋에 대한 finetuning은 3번과 동일하게 진행)

사전학습 시 Context의 토큰 길이 - 64, 엔티티 설명문의 길이 - 128
파인튜닝 시 Context의 토큰 길이 - 32, 엔티티 설명문의 길이 - 128

실험, 결과

- 데이터셋

표 3: 실험 성능 : Weak 매칭

Dataset	Precision	Recall	F1 Score
KT질의데이터셋	90.5	89.8	90.2
위키피디아	59.2	56.9	58.0
위키피디아 + KT질의데이터셋	93.4	93.6	93.5

표 4: 실험 성능 : Strong 매칭

Dataset	Precision	Recall	F1 Score
KT질의데이터셋	79.6	79.0	79.3
위키피디아	56.7	54.6	55.6
위키피디아 + KT질의데이터셋	86.1	86.3	86.2

결론

- 결론
  - 한국어 위키피디아 모든 엔티티(일부제외)를 포함한 시스템에서 단문질의에 대한 개체 연결을 잘 수행함을 알 수 있었고, 파인튜닝시 더 높은 성능을 얻을 수 있었다.
  - Bi-encoder를 이용함으로써 기존 모델보다 빠른 속도의 개체연결이 가능하다. (이는 실시간 질의응답 시스템에 중요한 부분이다.)

- 한계 및 향후 방향
  - KT질의데이터셋의 경우 문장 내 Seed Entity만을 연결하도록 만들어진 데이터셋이므로 추가적인 annotation을 통해 문장 내의 여러 개의 엔티티를 고려할 수 있는 데이터 셋으로 확장시킬 필요가 있다.
  - NIL을 고려하지 않았는데 NIL을 고려할 필요가 있다.
  - Graph Retriever(S. min et al. 2019)를 함께 사용하여 Open domain QA에 적용해볼 예정이다.

Q&A

감사합니다.