그래프⇋시퀀스의 반복적 추론을 이용한 한국어 AMR 파싱

민진우[O1], 나승훈[2], 최현수[3], 김한샘[4]

전북대학교[12], (주)엔씨소프트[3], 연세대학교[4]
Jinwoomin4488@gmail.com, nash@jbnu.ac.kr, choehyonsu@ncsoft.com, khss@yonsei.ac.kr

Korean AMR Parsing using Graph⇋Sequence Iterative Inference

Jinwoo Min[O1], Seung-Hoon Na[2], Hyonsu Choe[3], Young-Kil Kim[4]
Jeonbuk National University[12], NCSOFT Corp.[3], Yonsei University[4]

요 약

Abstract Meaning Representation(AMR)은 문장의 의미를 그래프 구조로 인코딩하여 표현하는 의미 형식 표현으로 문장의 각 노드는 사건이나 개체를 취급하는 개념으로 취급하며 간선들은 이러한 개념들의 관계를 표현한다. AMR 파싱은 주어진 문장으로부터 AMR 그래프를 생성하는 자연어 처리 태스크이다. AMR 그래프의 각 개념은 추상 표현으로 문장 내의 토큰과 명시적으로 정렬되지 않는 어려움이 존재한다. 이러한 문제를 해결하기 위해 별도의 사전 학습된 정렬기를 이용하여 해결하거나 별도의 정렬기 없이 Sequence-to-Sequence 계열의 모델로 입력 문장으로부터 그래프의 노드를 생성하는 방식으로 연구되어 왔다. 본 논문에서는 문장의 입력 시퀀스와 부분 생성 그래프 사이에서 반복 추론을 통해 새로운 노드와 기존 노드와의 관계를 구성하여 점진적으로 그래프를 구성하는 모델을 한국어 AMR 데이터 셋에 적용하여 Smatch 점수 39.8%의 실험 결과를 얻었다.

주제어: AMR 파싱, 그래프, Pointer Generator

1. 서론

Abstarct Meaning Representation(AMR) 파싱은 문장 단위의 의미(semantics)를 포착할 수 있도록 자연어 문장을 그래프 형태인 AMR로 변환하는 태스크이다. AMR은 루트, 방향성, 레이블이 존재하는 문장의 의미를 표현하는 그래프이며 그래프의 노드는 사건 혹은 개념(concept)를 나타내며 각 간선은 개념 사이의 관계(relation)을 표현한다[1]. AMR 파싱은 기존의 시멘틱 파싱과 달리 그래프의 노드와 문장의 토큰 사이의 명시적인 정렬이 존재하지 않고(no-anchored) 트리와의 대조적인 특성으로 다른 노드와 여러 관계를 맺는 재진입(reentrance)[3] 문제가 존재한다.

본 논문에서는 별도의 사전 학습된 정렬기(pretrained aligner)없이 입력 문장과 부분 생성 그래프 사이의 반복 추론을 통해 그래프를 완성해 나가는 시퀀스 그래프 반복 추론 모델[4]을 한국어 AMR 데이터 셋에 적용하여 실험결과 반복 추론의 효과를 보였으며 최대 Smatch 점수 39.8%를 달성하였다.

2. 관련 연구

영문 AMR 파싱은 입력 문장 내의 토큰과 그래프의 노드 사이의 정렬(alignment)을 학습한 사전 학습된 정렬기를 이용하여 그래프를 생성하는 방식과 별도의 정렬기 없이 그래프를 생성하는 두가지 방식으로 나뉜다. [2]에서는 학습데이터로부터 입력 토큰과 그래프 노드 사이의 정렬기를 학습한 후 전이 기반 방식을 이용하여 AMR 파싱을 수행하였으며 [11]는 정렬기를 잠재 변수로 취급하여 개념, 관계, 정렬을 모두 통합하여 모델링하는 방식을 제안하였다.

[3]는 별도의 정렬기의 학습 없이 2단계로 입력 시퀀스를 그래프로 변환하는 모델을 제안하였으며 위 방식은 먼저, 재진입 문제를 해결하기 위해서 입력 토큰 뿐 아니라 미리 생성된 그래프 노드도 복사하는 확장된 Pointer Generator[7]모델을 제안하였으며 재진입된 노드는 한번 더 생성하는 방식으로 해결하여 미리 생성된 그래프로부터 복사된 노드는 상호참조(coreference)된 상태로 표현한다. 다음으로, Pointer Generator를 이용하여 생성된 노드들에 대해서 Biaffine 어텐션[10] 모델을 적용하여 루트를 가지는 파스 트리를 얻고 상호 참조 정보를 이용하여 트리를 최종적인 AMR을 그래프로 하는 방식으로 AMR 파싱을 수행하는 모델이다.

[4]는 입력 시퀀스와 그래프 사이의 상호 정보를 주고받는 반복 추론을 통해 각각 표상을 얻은 후 이를 이용하여 그래프의 노드를 생성하며 관계를 예측하는 모델을 제안한다. 이 방식에서는 모든 노드를 생성한후 관계를 예측하여 그래프를 생성하는 것이 아닌 노드가 생성될 때마다 부분 생성된 그래프 사이의 관계를 생성하는 점진적 그래프 생성 방식을 제안한다. 또한, 생성된 노드

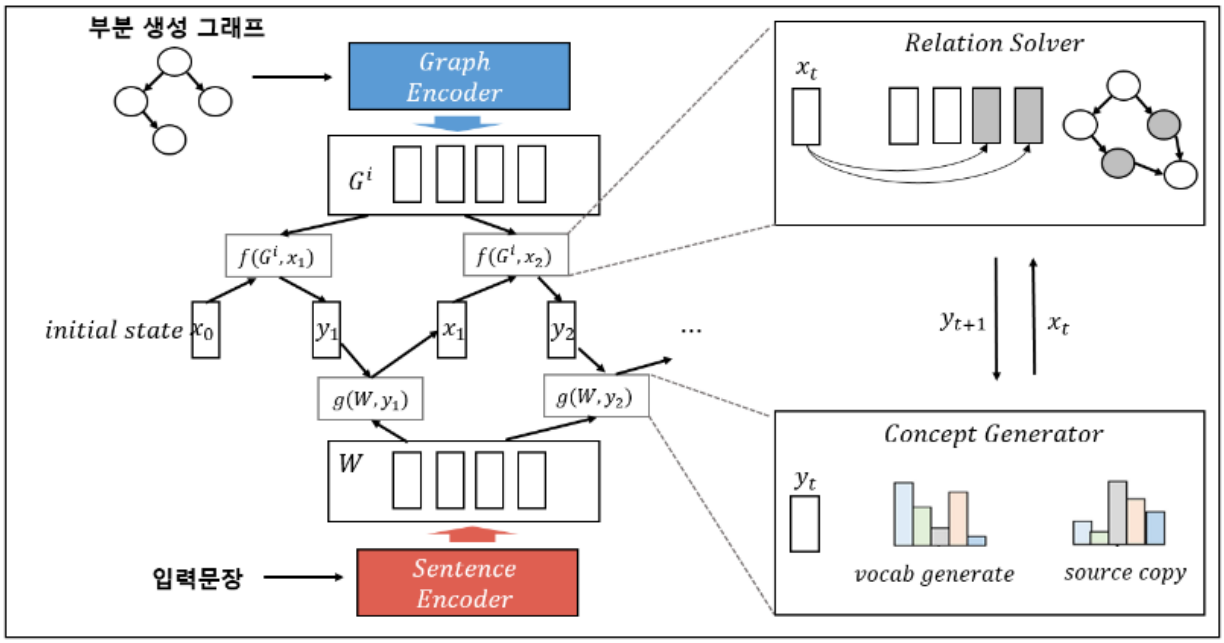

그림 1. 시퀀스⇋그래프 반복 추론 모델의 구조

에 대해 여러 개의 헤드를 두고 어텐션을 통해 기존 노드와의 관계를 해결하는 방식으로 재진입 문제를 해결하는 점에서 [3]와의 차이점이다.

한국어 의미 표상 체계 구축을 위해 [12]에서는 AMR에 대한 개념 및 체계 소개와 함께 한국어 AMR 자원 구축을 위한 고려사항 및 어려움 등에 대한 분석을 수행하였고 [6]의 연구는 이를 토대로 영문 현행 AMR 1.2.6 가이드라인을 한국어의 특성에 맞게 로컬라이징하여 한국어 AMR 가이드라인 1.0을 제시한 후 데이터 셋을 구축하였다.

3. 모델

본 모델은 [4]의 시퀀스⇋그래프 반복 추론 모델을 한국어에 적용할 수 있도록 변형한 모델로 문장 인코더의 입력 임베딩을 제외한 나머지 구조는 [4]와 동일하다. 초기의 그래프 G^0에서부터 그래프를 점진적으로 확장하여 최종 그래프를 얻는 $\{G^0 \rightarrow \cdots \rightarrow G^i \rightarrow \cdots\}$ 일련의 시리즈로 구성되며 그림 1에서 보듯이 각 그래프 확장 단계에서 시퀀스⇋그래프 반복 추론 절차가 진행된다.

$$y_t^i = g(G^i, x_t^i)$$
$$x_{t+1}^i = f(W, y_t^i)$$

그림에서 보듯이 (1) 주어진 입력 문장 내의 토큰을 인코딩하여 텍스트 메모리 W를 생성하는: Sentence 인코더, (2) 그래프 노드를 인코딩하여 부분 그래프 메모리 G^i를 생성하는 Graph 인코더, (3) 그래프의 새로운 개념 노드를 생성하는 Concept Generator, (4)새로운 노드와 기존 그래프 노드 사이의 관계를 결정하는 Relation Solver 4개의 파트로 구성되어 있다. 그림에서 보듯이 텍스트 메모리는 주어진 문장에 대해 인코딩하며 고정된 형태로 주어지고 그래프 메모리는 파싱 절차를 통해 점진적으로 변화됨을 알 수 있다.

3.1 Sentence 인코더

입력 문장에 대해서 임베딩 벡터를 얻고 이를 인코더를 통해 인코딩하는 과정을 거치며 입력 임베딩 이용되는 정보는 다음과 같다. 한국어 AMR 파싱을 위한 입력문장의 토큰 단위는 형태소로 입력 단계에서 형태소, 품사 태그, 개체명 태그와 형태소를 구성하는 음절들을 CNN(Convolutional Neural Networks)로 합성하여 얻어진 벡터를 결합하여 얻어진다. 추가적으로, 입력 형태소를 사전 학습된 한국어 BERT[9]모델을 통해 얻어진 표상을 추가로 포함한다.

이렇게 얻어진 입력 임베딩 시퀀스를 $\{w_0, w_1, \cdots w_n\}$에서 표현하며 w_0는 문장의 시작을 나타내는 BOS 토큰이다. 인코더로 RNN 계열의 인코더가 아닌 다층의 트랜스포머 [8]를 사용하며 입력 임베딩 시퀀스를 인코딩한 문장 시퀀스는 $\{h_0, h_1, \cdots h_n\}$로 표현하며 BOS 위치의 h_0는 문장의 전체적인 문맥에 대한 정보가 담겨있다.

3.2 Graph 인코더

그래프는 단순히 개념 노드들의 시퀀스로 표현하여 점진적으로 증가하는 그래프를 인코딩한다. 1) masked 셀프 매칭 어텐션 2) source 어텐션의 두가지의 어텐션을 사용하는 [8]의 트랜스포머 디코더와 동일한 형태로 Masked 셀프 매칭 어텐션은 점진적으로 생성된 그래프에서 생성된 노드위치까지만 어텐션이 수행되도록 제한하도록 셀프 매칭 어텐션을 수행하고 source 어텐션은 인코딩된 문장 시퀀스 $\{h_0, h_1, \cdots h_n\}$와의 어텐션을 수행한다.

그래프 인코더의 입력 임베딩은 개념 임베딩, 개념을 구성하는 음절들을 CNN을 통해 얻어진 벡터를 연결하여 얻어진다. 부분 그래프 임베딩 시퀀스 $\{c_0, c_1, c_2, \cdots c_m\}$를 Graph 인코더를 통해 인코딩하여 $\{s_0, s_1, s_2, \cdots s_m\}$을 얻으

며 여기서 m은 현재까지 생성된 부분 그래프의 길이를 나타내고 s_0는 빈 그래프를 표현하는 dummy 노드이다.

3.3 Concept Generator

t번의 반복 추론 과정을 거친 후 Concept Generator에서는 graph decision을 나타내는 상태 벡터 y_t와 입력 시퀀스 메모리 $h_{1:n}$를 받아 다음 수식과 같이 어텐션을 수행하여 모든 입력 시퀀스에 대한 어텐션 분포를 얻는다.

$$\mathrm{a}_t = \mathrm{softmax}(\frac{(W^Q y_t)^T W^K h_{1:n}}{\sqrt{d_k}})$$

얻어진 a_t는 새로운 개념과 기존 입력 시퀀스 사이의 가중치를 나타내며 이를 MLP를 적용한 후 미리 정의된 단어장에 대한 생성 확률 $p^{(vocab)}$을 얻는다.

$$\mathrm{MLP}(a_t) = (W^V h_{1:n})a_t + y_t$$
$$p^{(vocab)} = softmax(W^{(vocab)}\mathrm{MLP}(a_t) + b^{(vocab)})$$

또한 a_t는 생성될 노드를 입력 시퀀스의 형태소로부터 복사하는 확률로써 이용하는 복사 메커니즘으로 이용될 수 있다. Pointer Generator[7]와 같이 p_0, p_1 2개의 채널을 두고 최종적인 노드 예측에 대한 확률 $P(c)$은 다음 수식과 같이 표현된다.

$$[p_0, p_1] = softmax(W^{(switch)}\mathrm{MLP}(a_t))$$
$$P(c) = p_0 \cdot p^{(vocab)}(c) + p_1 \cdot (\sum_{i \in M(c)} a_t[i])$$

영문에서는 lemma, token의 두가지 시퀀스로부터 복사하기 위해 3가지 채널을 이용하지만 한국어는 형태소 단위로만 입력을 구성하고 복사 또한 하나의 시퀀스로부터 이루어지기 때문에 2가지 채널을 이용하여 노드 예측 분포를 구성한다.

3.4 Relation Solver

t번의 추론 과정을 거친 후 개념 결정 상태 벡터 y_t와 그래프 인코더를 통해 현재까지 생성된 그래프 메모리를 $s_{0:m}$를 이용해. Relation Solver에서는 새롭게 생성될 노드와 기존 노드 사이의 관계가 존재하는지를 판별하는 관계 인식 단계와 인식된 간선에 대해 해당하는 레이블을 부착하는 관계 분류의 두 단계로 분리된다. AMR은 트리와 달리 그래프의 구조로 표현하며 하나의 개념 노드가 여러 노드와 관계를 갖는 재진입을 허용한다. 여러 관계를 갖는 노드에 대해서 해당 노드를 복사하고 복사된 노드에 대한 상호 참조 정보를 저장하는 확장된 Pointer Generator를 통해 해결한 [3]과 달리 관계 인식 단계에서 멀티 헤드 어텐션을 사용하여 여러 개의 동시에 결정한다. 총 H개의 다른 헤드를 구성하며 각 헤드 h에 대해서 관계를 결정한다.

$$\beta_t^h = \mathrm{softmax}(\frac{(W_h^Q y_t)^T W_h^K s_{0:m}}{\sqrt{d_k}})$$

관계 분류 단계에서는 Biaffine 어텐션을 단순 적용하여 해당 간선에 대한 레이블을 결정한다.

3.5 Iterative Inference

먼저 초기 단계에서 문장의 모든 정보를 담겨있는 h_0로 x_0를 초기화하며 3.3절의 Concept Generator와 3.4절의 Relation Solver에서 각각 입력 시퀀스와 그래프에 대해서 어텐션을 수행하여 얻어진 어텐션 가중치 a_t, β_t^h를 이용하여 다음 함수를 통해 y_t, x_t를 업데이트한다.

$$y_t = f(G^i, x_{t-1}) = FFN^{(y)}(x_{t-1} + (W^V h_{1:n})\mathrm{a}_t)$$
$$x_t = g(W, y_t) = FFN^{(x)}(y_t + \sum_{h=1}^{H}(W_h^V s_{0:m})\beta_t^h)$$

최대 반복 횟수 N번의 반복과정을 거쳐 최종적으로 x_N, y_N이 얻어질 때까지 반복 한 후 이를 이용하여 새로운 노드 생성과 생성된 노드와 기존 노드들간의 관계를 결정한다.

$$x_0 \rightarrow f(G^i, x_0) \rightarrow y_1 \rightarrow g(W, y_1) \rightarrow x_1 \rightarrow \cdots$$
$$\rightarrow f(G^i, x_{N-1}) \rightarrow y_N \rightarrow g(W, y_N) \rightarrow x_N$$

새롭게 생성된 노드가 그래프의 끝을 알리는 EOG이면 최종적인 그래프를 반환하고 그렇지 않으면 현재까지 구성된 그래프 G^i에 새롭게 생성된 노드와 인식 & 분류된 관계들을 추가한 그래프 G^{i+1}로 확장한다.

4. 실험

4.1 실험 세팅

실험 및 평가를 위해 본 논문에서는 [6]의 한국어 AMR데이터 셋을 사용한다. 이 데이터 셋은 영문 AMR 1.2.6를 한국어의 특성에 맞게 로컬라이징한 데이터 셋으로 총 문장으로 1261문장 구성되며 각각 999 문장의 학습 셋과 126문장의 개발셋, 126 문장의 평가 셋으로 구성되어 있다. 아래의 표 1은 모델에 사용된 하이퍼 파라미터 세팅을 보여준다.

표 1. 모델의 하이퍼 파라미터

	Hyper Parameter	값
Sentence 인코더	형태소 임베딩 차원	300
	음절 임베딩 차원	50
	CNN filter 크기	100
	품사 임베딩 차원	50
	개체명 임베딩 차원	50
	레이어 수	1
	dropout 비율	0.1
Graph 인코더	개념 임베딩 차원	300
	레이어 수	1
	dropout 비율	0.1
Concept Generator	Feed-forward 은닉 차원 수	512
	Concept 단어장 크기	1242
Relation Solver	Feed-forward 은닉 차원 수	512
	헤드 수	8
모델	총 에포크	100
	배치 사이즈	32
	Optimizer	Adam
	학습율	$5e^{-4}$
	(Beta1, Beta2)	(0.9,0.999)

4.2 실험 결과

비교 실험을 위해 제안 모델과 입력 자질을 동일하게 설정한 STOG(Sequence To Graph Transduction)[3]을 한국어 동일 셋에 적용한 결과를 제시하며 또한 본 모델에서 최대 반복 추론 횟수에 따른 성능 변화를 표기한다. 성능 지표로는 생성된 그래프가 정답 AMR 그래프 대비 의미 관계를 잘 나타내는지에 대한 Smatch[5] 점수를 이용한다.

표 2. AMR 파싱 실험 결과

	F1
STOG	44.1%
Graph ≒ Sequence(Iteration 1)	33.6%
Graph ≒ Sequence(Iteration 2)	39.4%
Graph ≒ Sequence(Iteration 3)	39.0%
Graph ≒ Sequence(Iteration 4)	**39.8%**
Graph ≒ Sequence(Iteration 5)	39.2%

표 2에서 보듯이 최대 반복 횟수 4일때 가장 높은 성능을 보여주고 있으며 반복 횟수가 1에서 2로 변화될 때 성능 향상 폭이 크며 2 이상의 반복 횟수에서는 반복횟수 증가에 따른 따른 성능 변화가 미미한 결과를 보여준다. [4]의 결과에서처럼 영문에서는 Graph ≒ Sequence모델이 기존의 STOG에 비해 높은 성능을 보여주고 있으나 현재 한국어 셋 실험상에선 약 4.3%가량 낮은 성능을 보여주고 있다. 현재 구축된 한국어 AMR 데이터의 학습셋은 총 999문장으로 영문 AMR 데이터 셋이 36,524 문장에 비해 현저히 적은 양으로 구축되어 있어 모델 간의 비교를 위해 증강된 한국어 AMR셋에서의 성능 평가가 필요하다.

5. 결론

본 연구에서는 입력 시퀀스와 부분 생성된 그래프 사이의 반복 추론 모델을 한국어 AMR 파싱 데이터 셋에 적용하여 반복 추론의 효과를 보였으며 Smatch 점수 39.8%의 성능을 달성하였다.향후 연구로는 증강된 한국어 AMR 데이터 셋에 대한 실험을 진행하고 반복 추론 과정을 통해 얻어진 표상에 대해서 Seuence To Graph Transduction의 그래프 생성 방식을 적용한 하이브리드 모델에 대한 연구를 진행할 예정이다.

참고문헌

[1] Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., ... & Schneider, N. (2013, August). Abstract meaning representation for sembanking. In Proceedings of the 7th linguistic annotation workshop and interoperability with discourse (pp. 178-186).

[2] Damonte, M., Cohen, S. B., & Satta, G. (2016). An incremental parser for abstract meaning representation. arXiv preprint arXiv:1608.06111.

[3] Zhang, S., Ma, X., Duh, K., & Van Durme, B. (2019). Amr parsing as sequence-to-graph transduction. arXiv preprint arXiv:1905.08704.

[4] Cai, D., & Lam, W. (2020). Amr parsing via graph-sequence iterative inference. arXiv preprint arXiv:2004.05572.

[5] Cai, S., & Knight, K. (2013, August). Smatch: an evaluation metric for semantic feature structures. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) (pp. 748-752).

[6] 최현수, 한지윤, 박혜진, 오태환, 박석원, 김한샘 (2019). 문장 의미의 그래프 구조 표상을 위한 한국어 Abstract Meaning Representation 가이드라인. 제31회 한글 및 한국어 정보처리 학술대회 논문집

[7] See, A., Liu, P. J., & Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. arXiv preprint arXiv:1704.04368.

[8] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).

[9] http://aiopen.etri.re.kr/service_dataset.php

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

[11] Lyu, C., & Titov, I. (2018). Amr parsing as graph prediction with latent alignment. arXiv preprint arXiv:1805.05286.

[12] 최현수, 한지윤, 오태환, & 김한샘. (2019). 한국어 추상 의미 표상 (AMR) 을 위한 기초 연구. 언어, 44(4), 943-969.