# 제한된 타입 시그니처 기반의 시맨틱 파싱

남대환[1O], 이근배[1,2†]

포항공과대학교
{dhnam, gblee}@postech.ac.kr

# Semantic parsing with restricted type signatures

Daehwan Nam[1O], Gary Geunbae Lee[1,2†]
Department of Computer Science and Engineering, Pohang University of Science and Technology[1]
Graduate school of artificial intelligence, Pohang University of Science and Technology[2]

**요 약**

시맨틱 파싱은 주어진 자연어 발화를 domain specific language(DSL)를 따르는 프로그램으로 변환하는 방법이다. 시맨틱 파서가 다형성을 가지는 DSL을 사용할 경우, 적은 수의 토큰으로 다양한 프로그램을 출력할 수 있지만, 탐색 공간이 넓어진다는 문제가 있다. 본 연구에서는 해당 문제를 완화하기 위해 다형성을 가지는 DSL의 타입 시그니처를 제한하는 방법을 제안한다. 해당 방법은 sequence-to-sequence 기반의 시맨틱 파싱 성능을 향상시키는데 효율적임을 보였다.

주제어: 시맨틱 파싱, 질의 응답, sequence-to-sequence

## 1. 서론

실행 가능한 시맨틱 파싱(executable semantic parsing)은 주어진 자연어 발화를 대응되는 논리 형태(logical form) 또는 프로그램(program)으로 변환하는 방법이다. 시맨틱 파서가 출력한 프로그램은 정형 데이터(structured data) 또는 지식베이스(knowledge base)와 같은 문맥에 기반하여 사용자가 필요로 하는 정보 또는 결과를 출력할 수 있다. 따라서 시맨틱 파싱은 자연어를 통해 컴퓨터를 조작하는 인터페이스가 된다.

시맨틱 파서는 목표로 하는 태스크의 domain specific language(DSL)를 따르는 프로그램을 생성해야 한다. 주어진 태스크에 따라 DSL은 함수형 질의 언어[1], SQL 또는 범용 프로그래밍 언어[2]가 될 수 있다. 이러한 DSL과 일치하는 프로그램 생성을 위해서는 타입 또는 생성 규칙(production rule)에 기반한 문법이 사용된다. 또한 시맨틱 파싱의 성능은 DSL의 표현력(expressive power)과 프로그램 탐색 공간(search space)에 영향을 받는다. 예로, 다형성(polymorphism)이 포함된 DSL은 적은 수의 함수로 여러 프로그램을 조합 할 수 있다. 하지만, 표현력이 높아지면 탐색 공간 또한 넓어진다는 문제가 있다. 따라서 표현력과 탐색 공간이 절충된 DSL이 사용되어야 한다.

본 연구는 다형성을 가지는 DSL의 프로그램 탐색 공간 감소를 위해 타입 시그니처(signature)가 제한된 시맨틱 파싱을 제안한다. 제안된 방법은 Cornell Natural Language Visual Reasoning (NLVR)[7]의 structured representations 태스크에 적용되었으며, 시맨틱 파서의 성능을 높이는데 효과가 있음을 보였다.

## 2. 관련 연구

신경망 기반의 sequence-to-sequence(S2S) 모델이 각광을 받은 이후, S2S 기반의 시맨틱 파서가 제안되었다[3]. 하지만 S2S 기반 모델은 전통적인 모델[4]과 다르게 엄격한 타입 검사를 하지 않는다는 문제가 있었고, 이를 해결하기 위해 타입 또는 생성 규칙을 이용한 모델들이 제안되었다[2,5]. Krishnamurthy et al. [5] 의 연구는 시맨틱 파서의 행동(action)이 타입 유도(derivation)를 발생 시키는 생성 규칙으로, 연속된 행동에 의해 출력되는 최종 프로그램은 DSL를 따르게 된다. 본 연구는 Krishnamurthy et al. [5]의 연구와 유사하나, 각 행동이 프로그램을 구성하는 함수 또는 상수와 같은 토큰(token)에 대응된다는 차이가 있다.

## 3. 시맨틱 파싱

### 3.1. S2S 기반 시맨틱 파서

본 연구는 자연어 발화 $\boldsymbol{x} \in X$가 입력되었을 때, 대응되는 프로그램 $\boldsymbol{z} \in Z$를 출력하는 S2S 기반 시맨틱 파서 $f_\theta: X \rightarrow Z$ 를 학습하고자 한다. 해당 시맨틱 파서는 다음과 같이 정의 된다:

$$f_\theta(\boldsymbol{x}) = \underset{\boldsymbol{z} \in Z}{\operatorname{argmax}}\, p_\theta(\boldsymbol{z}|\boldsymbol{x})$$

$$p_\theta(\boldsymbol{z}|\boldsymbol{x}) = \Pi_{t=1}^{T} p_\theta(z_t|\boldsymbol{x}, \boldsymbol{z_{0:t-1}})$$

† 교신저자(corresponding author).

$\tilde{x}$: there is **[t-one]** tower with **[t-quantity-compare] [t-int] [t-color]** item

$\tilde{z}$: (exist (filter all-boxes (lambda (x) **([t-quantity-compare] [t-int]** (count (filter x (lambda (y) (is-**[t-color]** y))))))))

$x$: there is a single tower with exactly 2 blue item

$z$: (exist (filter all-boxes (lambda (x) (= 2 (count (filter x (lambda (y) (is-blue y))))))))

그림 1. 추상 예제와 생성된 구체적 예제. $\tilde{x}$는 추상 발화이고 $\tilde{z}$는 추상 프로그램이며, 이들로부터 구체적 발화 $x$와 구체적 프로그램 $z$가 생성된다.

| 프로그램 토큰 | 기본 타입 시그니처 | 제한된 타입 시그니처 |
|---|---|---|
| lambda | Bool → BoolFunc(T) | Bool → BoolFunc(Item)<br>Bool → BoolFunc(Set(Item)) |
| filter | [Set(T), BoolFunc(T)] → Set(T) | [Set(Item), BoolFunc(Item)] → Set(Item)<br>[Set(Set(Item)), BoolFunc(Item)] → Set(Set(Item)) |
| all | [Set(T), BoolFunc(T)] → Bool | [Set(Item), BoolFunc(Item)] → Bool<br>[Set(Set(Item)), BoolFunc(Set(Item))] → Bool |

표 1. 제한된 타입 시그니처 예시. 타입 시그니처에서 화살표 왼쪽은 매개변수 타입이며, 오른쪽은 반환 타입이다. 타입 변수를 포함하는 기본 타입 시그니처가 몇 개의 구체적 타입 시그니처로 제한되었다.

여기서 $p_\theta(z|x)$는 S2S로 학습된 확률 분포로, 각 프로그램 토큰 $z_t$가 나올 확률 $p_\theta(z_t|x, z_{0:t-1})$를 이용하여 계산된다. 해당 확률 분포 계산을 위해, LSTM 기반의 인코더와 디코더가 사용되었다.

시맨틱 파서가 조합한 프로그램 $z$는 주어진 문맥 $k$를 입력으로 실행결과 또는 디노테이션(denotation) $y$를 출력한다. 시맨틱 파서가 올바른 프로그램을 생성했다면, 디노테이션은 사용자의 발화를 만족시키는 값이 된다. 예를 들어, 그림 1의 프로그램 $z$가 실행 될 때, 객체 사이의 관계 정보를 포함하는 데이터가 문맥 $k$로서 전달되면, 해당 프로그램은 문맥에 따라 True 또는 False 라는 디노테이션 $y$를 출력한다.

우리는 해당 시맨틱 파서를 NLVR 태스크에 대해 학습하기 위해 Goldman et al. [6]이 제안한 106개의 추상 예제(abstract example)를 사용하였다. 추상 예제는 NLVR 훈련데이터에서 가장 빈도 높은 패턴을 수집하여 추상적인 발화 및 프로그램을 표기한 것이다. 추상 예제를 이용하면 이로부터 구체적인 예제를 생성할 수 있다. 예를 들어, 그림 1의 추상 예제 $\tilde{x}$와 $\tilde{z}$를 구성하는 심볼인 t-one, t-quantity-compare t-int t-color를 구체적인 발화 및 프로그램의 심볼로 대체하면 구체적 발화 $x$와 프로그램 $z$가 생성된다. 추상예제로부터 학습을 위해 다음 목적 함수가 사용된다:

$$\text{Loss}(\tilde{D}) = \sum_{(\tilde{x},\tilde{z})\in\tilde{D}} \log p_\theta(z|x)$$

$$\text{where } (x, z) \sim (\tilde{x}, \tilde{z})$$

$\tilde{D}$는 전체 추상 예제의 집합이며, 구체적 예제$(x, z)$는 추상 예제 $(\tilde{x}, \tilde{z})$로부터 샘플링 된다.

## 3.2. 타입 시그니처

프로그램을 구성하는 각 토큰의 타입 시그니처는 반환 타입을 가지며, 비단말(non-terminal) 토큰의 경우 1개 이상의 매개변수 타입을 포함한다. 시맨틱 파서는 프로그램 토큰을 추가 할 때 가장 최우측 개방된 비단말 토큰의 매개변수 타입과 새로 추가 될 토큰의 반환 타입이 일치하는지 검사해야 한다. 예를 들어, 현재까지 구축된 부분프로그램 $z_{0:t-1}$가 "(exist (filter all-boxes" 라고 하면 최우측 개방된 비단말 토큰은 filter가 되고, 다음 토큰은 filter의 두번째 인자에 대응되며, 해당 타입은 BoolFunc(T)가 되기 때문에 반환 타입이 일치하는 lambda가 다음 토큰으로 선택될 수 있다.

본 연구의 DSL은 Goldman et al. [6]와 같은 것으로, 타입표현은 Item, Color, Shape, Side와 같은 원시타입(primitive type), Set, BoolFunc와 같은 복합타입(composite type) 그리고 어떤 타입에도 대응 가능한 타입변수 T로 구성된다. 타입변수를 이용하면 매개변수 다형성(parametric polymorphism)을 가지는 토큰을 정의 할 수 있기 때문에 DSL의 표현력이 증가한다. 예를 들어 표 1의 filter 토큰은 Set(T)과 BoolFunc(T)의 인자를 받아 Set(T) 타입의 값을 반환하는 함수가 되며, T의 값은 Item 이나 Shape 등의 타입이 될 수 있다. 하지만 타입변수가 포함된 타입 시그니처는 여러 구체적 타입에 대응 될 수 있기 때문에 탐색 공간이 넓어진다는 문제가 있다

| 모델 | Acc.(dev.) | Con.(dev.) | Acc.(test-p) | Con.(test-p) |
|---|---|---|---|---|
| Goldman et al. [6] (Sup.) | 67.7 | 36.7 | 66.3 | 32.7 |
| Goldman et al. [6] (Sup.+Disc) | **77.7** | 52.4 | 76.6 | 51.8 |
| S2S | 76.0 | 52.0 | 75.1 | 50.3 |
| S2S + 시그니처 제한 | 76.8 | **54.8** | **77.0** | **55.2** |

표 2. 시맨틱 파서 성능 비교. S2S (+ 시그니처 제한) 모델은 평균 성능이 표기되었다.

## 4. 타입 시그니처 제한 방법

본 연구에서는 타입변수가 포함된 타입 시그니처를 최소한의 구체적 타입 시그니처 목록으로 변환하여 그 범위를 제한한다. 이를 위해 106개의 추상 예제의 프로그램이 사용되어, 각 토큰이 추상 프로그램 내에서 가질 수 있는 모든 타입 시그니처를 나열하였다. 예를 들어, 표 1의 filter의 타입 시그니처는 T값에 따라 여러 구체적 타입 시그니처에 대응 될 수 있지만, 106개의 추상 프로그램에서는 오직 2가지 경우만 발생한다. 이와 같은 과정을 모든 토큰에 대해서 수행하였다. 그 결과, 타입 변수가 시그니처에 포함되는 토큰 11개에서 총 21개의 구체적 타입 시그니처가 발생하는 것을 확인하였다.

제한된 수의 구체적 타입 시그니처가 정해지면, 각 토큰 마다 해당 시그니처의 수 만큼 중복되는 토큰을 만들고 시맨틱 파서가 이를 행동으로 선택할 수 있게 하였다. 예를 들어, filter 토큰의 경우 구체적 타입 시그니처가 2개이기 때문에 각 시그니처에 대응되는 filter-1과 filter-2를 만든다. 시그니처 제한 후, 전체 토큰 수는 51개에서 61개가 되었다. 이후 시맨틱 파서는 새로 만들어진 토큰을 순차적으로 출력하여 프로그램을 조합한다.

## 5. 실험

타입 시그니처 제한에 따른 2가지 시맨틱 파서를 학습하였다. 학습을 위해서, 추상 예제 106개가 사용되었으며, 각 추상 예제는 매번 구체적인 예제를 샘플링 하고, 이로부터 $\mathrm{Loss}(\tilde{D})$를 줄이도록 학습되었다. 각 모델은 10번씩 훈련되었으며 모델 instance 10개의 평균 성능이 측정 되었다. 성능 평가 척도인 accuracy는 각 발화를 입력으로 시맨틱 파서가 생성한 프로그램이 주어진 문맥 1개에 대해 올바른 디노테이션이 출력하는지를 의미하며, consistency는 동일한 프로그램이 주어진 서로 다른 문맥 4개에 대해 모두 올바른 디노테이션을 출력하는지를 의미한다. 훈련에 사용된 hyperparameter는 epoch 300회, batch 크기 8, 인코더 및 디코더 LSTM의 hidden state 크기 32와 48 등이 있다.

학습된 모델은 Goldman et al. [6]가 교사학습한 모델과 비교되었다 (표 2). 해당 모델은 feed-forward 디코더를 사용하는 구조로, 106개의 추상 예제로부터 만들어진 6,158 수의 구체적 예제로 학습되었다. 우선, 우리가 사용한 LSTM기반 S2S 모델은은 feed-fowrd 디코더 기반의 Sup. 모델보다 우수한 성능을 보인다. 그리고 타입 시그니처 제한이 적용된 S2S 모델은 Sup.+Disc 모델보다 높은 성능을 보인다. 본 방법이 성능을 향상시키는 원인은, 타입 시그니처가 제한되면서 불필요한 탐색 공간이 줄어들고, 탐색 공간에서 올바른 프로그램을 찾아야 하는 시맨틱 파서의 부담이 줄어들었기 때문으로 볼 수 있다.

## 6. 결론

본 연구는 다형성을 가지는 DSL의 타입 시그니처를 제한하여 시맨틱 파싱의 프로그램 탐색 공간을 줄이는 방법을 제안하였다. 해당 방법은 S2S 기반 시맨틱 파서에 적용되어 성능을 향상시켰다. 본 연구에서는 추상 예제를 활용하여 타입 시그니처를 제한하였으나, 구체적인 프로그램 예제를 이용하는 것도 가능할 것이다. 향후 계획은 약한지도학습 기반의 시맨틱 파싱에 적용하는 것과 프로그램 예제없이 자동으로 타입 시그니처를 제한하는 방법을 연구 할 것이다.

## 감사의 글

본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터지원사업의 연구결과로 수행되었음 (IITP-2020-2020-0-01789)

## 참고문헌

[1] Cheng, Jianpeng, et al. "Learning structured natural language representations for semantic parsing." Proceedings of ACL 2017.

[2] Yin, Pengcheng, and Graham Neubig. "A syntactic neural model for general-purpose code generation." Proceedings of ACL 2017.

[3] Dong, Li, and Mirella Lapata. "Language to logical form with neural attention." Proceedings of ACL 2016.

[4] Zettlemoyer, Luke S., and Michael Collins. "Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars." Proceedings of UAI 2005.

[5] Krishnamurthy, Jayant, Pradeep Dasigi, and Matt Gardner. "Neural semantic parsing with type constraints for semi-structured tables." Proceedings of EMNLP 2017.

[6] Goldman, Omer, et al. "Weakly-supervised semantic parsing with abstract examples." Proceedings of ACL 2018.

[7] Suhr, Alane, et al. "A corpus of natural language for visual reasoning." Proceedings of ACL. 2017.