Bringing machine learning & compositional semantics together: approaches

https://github.com/cgpotts/annualreview-complearning

Chris Potts
Stanford Linguistics

CS 244U: Natural language understanding

Semantic parsing

$$\langle \boxed{u, t, r}, d \rangle$$

Basic formulation

	Utterance	Logical form
Train	seven minus five	(− 7 5)
	five minus seven	(− 5 7)
	three plus one	(− 7 5)
	minus three plus one	(+ ¬3 1)
	minus three plus one	¬(+ 3 1)
	two minus two times two	(× (− 2 2) 2)
	two minus two times two	(− 2 (× 2 2))
	two plus three plus four	(+ 2 (+ 3 4))
	⋮	
Test	three minus one	?
	three times one	?
	minus six times four	?
	one plus three plus five	?
	⋮	

Table: Data requirements.

Basic formulation

	Utterance	Logical form
Train	seven minus five	(− 7 5)
	five minus seven	(− 5 7)
	three plus one	(− 7 5)
	minus three plus one	(+ ¬3 1)
	minus three plus one	¬(+ 3 1)
	two minus two times two	(× (− 2 2) 2)
	two minus two times two	(− 2 (× 2 2))
	two plus three plus four	(+ 2 (+ 3 4))
	⋮	
Test	three minus one	?
	three times one	?
	minus six times four	?
	one plus three plus five	?
	⋮	

Table: Data requirements.

Syntax	Logical form
N → one	1
N → one	2
	⋮
N → two	1
N → two	2
	⋮
R → plus	+
R → plus	−
R → plus	×
R → minus	+
R → minus	−
R → minus	×
R → times	+
R → times	−
R → times	×
S → minus	¬
N → S N	⌜S⌝⌜N⌝
N → N_L R N_R	(⌜R⌝ ⌜N_L⌝ ⌜N_R⌝)

Table: Crude grammar.

Learning framework

Learning framework

1. Feature representations: $\phi(x, y) \in \mathbb{R}^d$

Learning framework

1. Feature representations: $\phi(x, y) \in \mathbb{R}^d$
2. Scoring: $\text{Score}_{\mathbf{w}}(x, y) = \sum_{j=1}^{d} w_j \phi(x, y)_j$

Learning framework

1. Feature representations: $\phi(x, y) \in \mathbb{R}^d$
2. Scoring: $\text{Score}_{\mathbf{w}}(x, y) = \sum_{j=1}^{d} w_j \phi(x, y)_j$
3. Multiclass hinge-loss objective function:

$$\min_{\mathbf{w} \in \mathbb{R}^d} \sum_{(x,y) \in \mathcal{D}} \max_{y' \in \textsc{Gen}(x)} [\text{Score}_{\mathbf{w}}(x, y') + c(y, y')] - \text{Score}_{\mathbf{w}}(x, y)$$

where $\mathcal{D}$ is a set of (x, y) training examples and $c(a, b) = 1$ if $a \neq b$, else 0.

Learning framework

1. Feature representations: $\phi(x, y) \in \mathbb{R}^d$
2. Scoring: $\text{Score}_{\mathbf{w}}(x, y) = \sum_{j=1}^{d} w_j \phi(x, y)_j$
3. Multiclass hinge-loss objective function:

$$\min_{\mathbf{w} \in \mathbb{R}^d} \sum_{(x,y) \in \mathcal{D}} \max_{y' \in \textsc{Gen}(x)} [\text{Score}_{\mathbf{w}}(x, y') + c(y, y')] - \text{Score}_{\mathbf{w}}(x, y)$$

where $\mathcal{D}$ is a set of (x, y) training examples and $c(a, b) = 1$ if $a \neq b$, else 0.

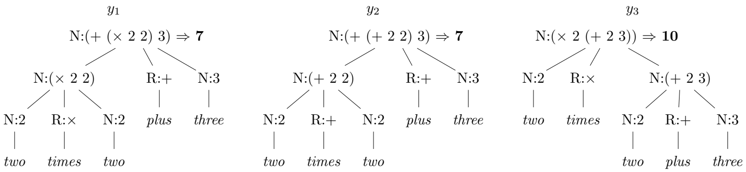
4. Optimization:

$\textsc{StochasticGradientDescent}(\mathcal{D}, T, \eta)$

```
Initialize w ← 0
Repeat T times
    for each (x, y) ∈ D (in random order)
        ỹ ← arg max_{y'∈GEN(x)} Score_w(x, y') + c(y, y')
        w ← w + η(φ(x, y) − φ(x, ỹ))
Return w
```

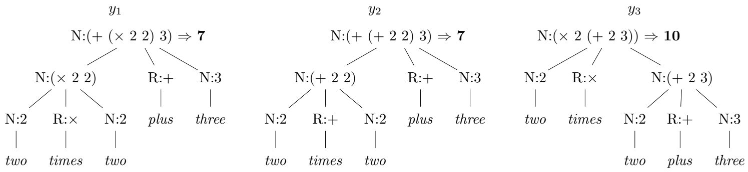
Example

(a) Candidates GEN(x) for utterance $x =$ *two times two plus three*



Example

(a) Candidates GEN(x) for utterance $x =$ *two times two plus three*



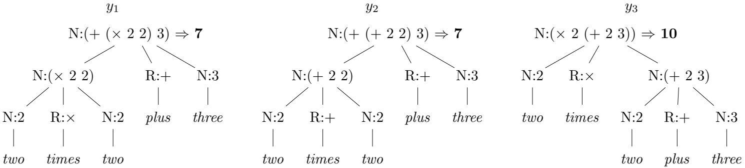
$\phi(x, y_1) = \begin{bmatrix} \text{R:}\times[\textit{times}] : 1 \\ \text{R:+}[\textit{plus}] : 1 \\ \text{top[R:+]} : 1 \end{bmatrix}$

$\phi(x, y_2) = \begin{bmatrix} \text{R:+}[\textit{times}] : 1 \\ \text{R:+}[\textit{plus}] : 1 \\ \text{top[R:+]} : 1 \end{bmatrix}$

$\phi(x, y_3) = \begin{bmatrix} \text{R:}\times[\textit{times}] : 1 \\ \text{R:+}[\textit{plus}] : 1 \\ \text{top[R:}\times\text{]} : 1 \end{bmatrix}$

Example

(a) Candidates GEN(x) for utterance $x =$ *two times two plus three*



$\phi(x, y_1) =$ R:×[*times*] : 1, R:+[*plus*] : 1, top[R:+] : 1

$\phi(x, y_2) =$ R:+[*times*] : 1, R:+[*plus*] : 1, top[R:+] : 1

$\phi(x, y_3) =$ R:×[*times*] : 1, R:+[*plus*] : 1, top[R:×] : 1

(b) Learning from logical forms (Section 4.1)

Iteration 1

$\mathbf{w} =$ R:×[*times*] : 0, R:+[*times*] : 0, R:+[*plus*] : 0, top[R:+] : 0, top[R:×] : 0

Scores: $[0, 0, 0]$
$y = y_1$
$\tilde{y} = y_3$ (tied with y_2)

⇒

Iteration 2

$\mathbf{w} =$ R:×[*times*] : 0, R:+[*times*] : 0, R:+[*plus*] : 0, top[R:+] : 1, top[R:×] : -1

Scores: $[1, 1, -1]$
$y = y_1$
$\tilde{y} = y_2$

⇒

Iteration 3

$\mathbf{w} =$ R:×[*times*] : 1, R:+[*times*] : -1, R:+[*plus*] : 0, top[R:+] : 1, top[R:×] : -1

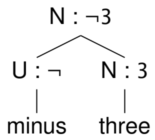
Scores: $[2, 0, 0]$
$y = y_1$
$\tilde{y} = y_1$

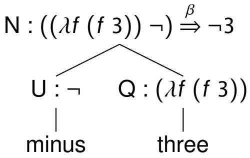
Derivational ambiguity

Syntax	Logical form
N → one	1
N → two	2
	⋮
R → plus	$+$
R → minus	$-$
R → times	$\times$
S → minus	$\neg$
N → S N	$\ulcorner S \urcorner \ulcorner N \urcorner$
N → N_L R N_R	$(\ulcorner R \urcorner\ \ulcorner N_L \urcorner\ \ulcorner N_R \urcorner)$
Q → *n*	$(\lambda f\ (f\ \ulcorner n \urcorner))$
N → U Q	$(\ulcorner Q \urcorner\ \ulcorner U \urcorner)$

Table: Grammar with type-lifting.

Training instance: (*minus three*, $\neg 3$)





(Beta-conversion $\overset{\beta}{\Rightarrow}$ is the syntactic counterpart of functional application.)

Derivations as latent variables

- The training instances are (u, r) pairs.
- Since r might have multiple derivations, derivations are latent variables.
- Latent support vector machine objective:

$$\min_{\mathbf{w}\in\mathbb{R}^d} \sum_{(x,r)\in\mathcal{D}} \max_{y'\in\textsc{Gen}(x)} [\text{Score}_{\mathbf{w}}(x, y') + c(r, \textsc{Root}(y'))] - \max_{y''\in\textsc{Gen}(x,r)} \text{Score}_{\mathbf{w}}(x, y''),$$

where $\mathcal{D}$ is a set of (utterance, formula) pairs; $c(a, b) = 1$ if $a \neq b$, else 0; and $\textsc{Gen}(x, r) = \{y \in \textsc{Gen}(x) : \textsc{Root}(y) = r\}$

- Optimization:

$\textsc{StochasticGradientDescent}(\mathcal{D}, T, \eta)$

Initialize $\mathbf{w} \leftarrow \mathbf{0}$
Repeat T times
    for each $(x, r) \in \mathcal{D}$ (in random order)
        $y \leftarrow \arg\max_{y''\in\textsc{Gen}(x,r)} \text{Score}_{\mathbf{w}}(x, y'')$
        $\tilde{y} \leftarrow \arg\max_{y'\in\textsc{Gen}(x)} \text{Score}_{\mathbf{w}}(x, y') + c(y, y')$
        $\mathbf{w} \leftarrow \mathbf{w} + \eta(\phi(x, y) - \phi(x, \tilde{y}))$
Return $\mathbf{w}$

Learning from denotations

$$\langle \boxed{u, t, r, d} \rangle$$

Motivations

Semantic parsing

- What is the largest city in California?
- $\arg\max\left(\{c : \text{city}(c) \wedge \text{loc}(c, \text{CA})\}, \text{population}\right)$

Interpretive

- What is the largest city in California?
- Los Angeles.

Basic formulation

	Utterance	Denotation
	seven minus five	2
	five minus seven	−2
	three plus one	4
	minus three plus one	−2
Train	minus three plus one	−4
	two minus two times two	0
	two minus two times two	−2
	two plus three plus four	9
	⋮	
	three minus one	?
	three times one	?
Test	minus six times four	?
	one plus three plus five	?
	⋮	

Table: Data requirements.

Basic formulation

	Utterance	Denotation
Train	seven minus five	2
	five minus seven	−2
	three plus one	4
	minus three plus one	−2
	minus three plus one	−4
	two minus two times two	0
	two minus two times two	−2
	two plus three plus four	9
	⋮	
Test	three minus one	?
	three times one	?
	minus six times four	?
	one plus three plus five	?
	⋮	

Table: Data requirements.

Syntax	Logical form	Denotation
N → one	1	1
N → one	2	2
	⋮	
N → two	1	1
N → two	2	2
	⋮	
R → plus	$+$	addition
R → plus	$-$	subtraction
R → plus	$\times$	multiplication
R → minus	$+$	addition
R → minus	$-$	subtraction
R → minus	$\times$	multiplication
R → times	$+$	addition
R → times	$-$	subtraction
R → times	$\times$	multiplication
S → minus	$\neg$	negative
N → S N	$\ulcorner S \urcorner \ulcorner N \urcorner$	$[\![\ulcorner S \urcorner]\!]([\![\ulcorner N \urcorner]\!])$
N → N_L R N_R	$(\ulcorner R \urcorner\ \ulcorner N_L \urcorner\ \ulcorner N_R \urcorner)$	$[\![\ulcorner R \urcorner]\!]([\![\ulcorner N_L \urcorner]\!], [\![\ulcorner N_R \urcorner]\!])$

Table: Crude grammar.

Learning framework

Feature representations and scoring are as before.

1. Latent support vector machine objective:

$$\min_{\mathbf{w}\in\mathbb{R}^d} \sum_{(x,d)\in\mathcal{D}} \max_{y'\in\text{Gen}(x)} [\text{Score}_{\mathbf{w}}(x, y') + c(d, [\![y']\!])] - \max_{y\in\text{Gen}(x,d)} \text{Score}_{\mathbf{w}}(x, y),$$

   where $\text{Gen}(x, d) = \{y \in \text{Gen}(x) : [\![y]\!] = d\}$ is the set of logical forms that evaluate to denotation d.

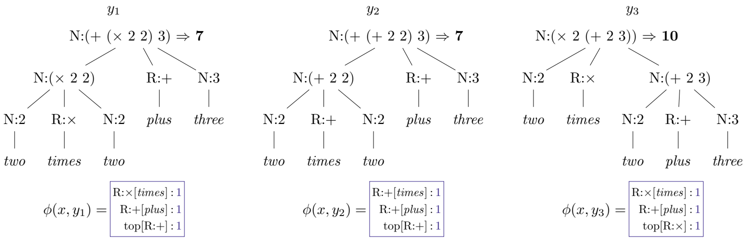
2. Optimization:

   $\text{StochasticGradientDescent}(\mathcal{D}, T, \eta)$

   Initialize $\mathbf{w} \leftarrow \mathbf{0}$
   Repeat T times
       for each $(x, d) \in \mathcal{D}$ (in random order)
           $y \leftarrow \arg\max_{y''\in\text{GEN}(x,d)} \text{Score}_{\mathbf{w}}(x, y'')$
           $\tilde{y} \leftarrow \arg\max_{y'\in\text{GEN}(x)} \text{Score}_{\mathbf{w}}(x, y') + c(y, y')$
           $\mathbf{w} \leftarrow \mathbf{w} + \eta(\phi(x, y) - \phi(x, \tilde{y}))$
   Return $\mathbf{w}$

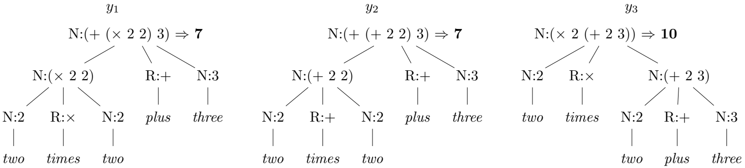
Example

(a) Candidates GEN(x) for utterance $x =$ *two times two plus three*



Example

(a) Candidates GEN(x) for utterance $x =$ *two times two plus three*



$\phi(x, y_1) =$ R:×[*times*] : 1, R:+[*plus*] : 1, top[R:+] : 1

$\phi(x, y_2) =$ R:+[*times*] : 1, R:+[*plus*] : 1, top[R:+] : 1

$\phi(x, y_3) =$ R:×[*times*] : 1, R:+[*plus*] : 1, top[R:×] : 1

(c) Learning from denotations (Section 4.2)

Iteration 1

$\mathbf{w} =$ R:×[*times*] : 0, R:+[*times*] : 0, R:+[*plus*] : 0, top[R:+] : 0, top[R:×] : 0

Scores: $[0, 0, 0]$
GEN$(x, d) = \{y_1, y_2\}$
$y = y_1$ (tied with y_2)
$\tilde{y} = y_3$

⇒

Iteration 2

$\mathbf{w} =$ R:×[*times*] : 0, R:+[*times*] : 0, R:+[*plus*] : 0, top[R:+] : 1, top[R:×] : -1

Scores: $[1, 1, -1]$
GEN$(x, d) = \{y_1, y_2\}$
$y = y_1$ (tied with y_2)
$\tilde{y} = y_1$ (tied with y_2)