Sequence-to-Action: End-to-End Semantic Graph Generation for Semantic Parsing

Bo Chen, Le Sun, Xianpei Han

紹介者:
名古屋大学大学院工学研究科
情報・通信工学専攻 佐藤・松崎研究室
修士2年 犬塚慎也

目次

1. 意味解析 (Semantic Parsing) とは
2. 意味解析の手法
3. 論文の概要，提案手法の概略

4. 意味グラフを構築する操作の設計，構築例
5. Seq2Act モデルの設計，構造的・意味的制約

6. 評価・エラー分析
7. まとめ

意味解析 (Semantic Parsing) とは

- 自然言語の入力文を論理式にマッピングする
  - 質問応答タスクのための手法の1つ
  - 入力文の構造予測と意味付けの2つの機能が必要

"Which states border Texas?"

入力: 質問文

意味解析 (Semantic Parsing)

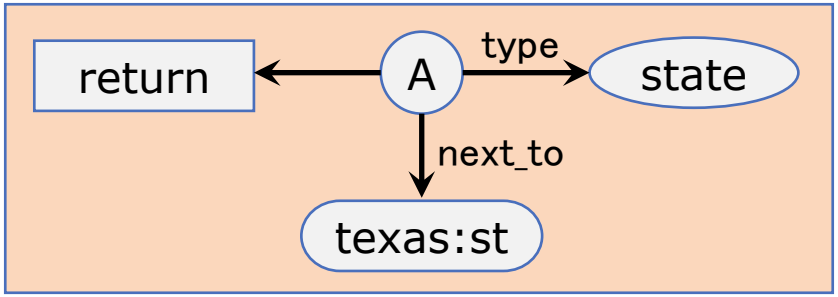
```
answer(A, (
    state(A),
    next_to(A, stateid(texas))
))
```

出力: 論理式

(論理形式, logical form)

意味解析の手法

1. 構成的言語理論 (CCG, DCS, …) に基づく手法 [Liang et al., 2011]他
   - 文法で入力文の構造を決め，語彙項目で意味付け
   - 広いドメインでは文法の設計などが困難，end-to-end ではない
2. 意味グラフ (Semantic Graph) に基づく手法 [Reddy et al., 2014]他
   - 知識ベースへのクエリを表現したグラフを出力
   - 知識ベースとのマッチングなどのヒューリスティクスに基づく
     → 広いドメインや複雑な文では対処が困難



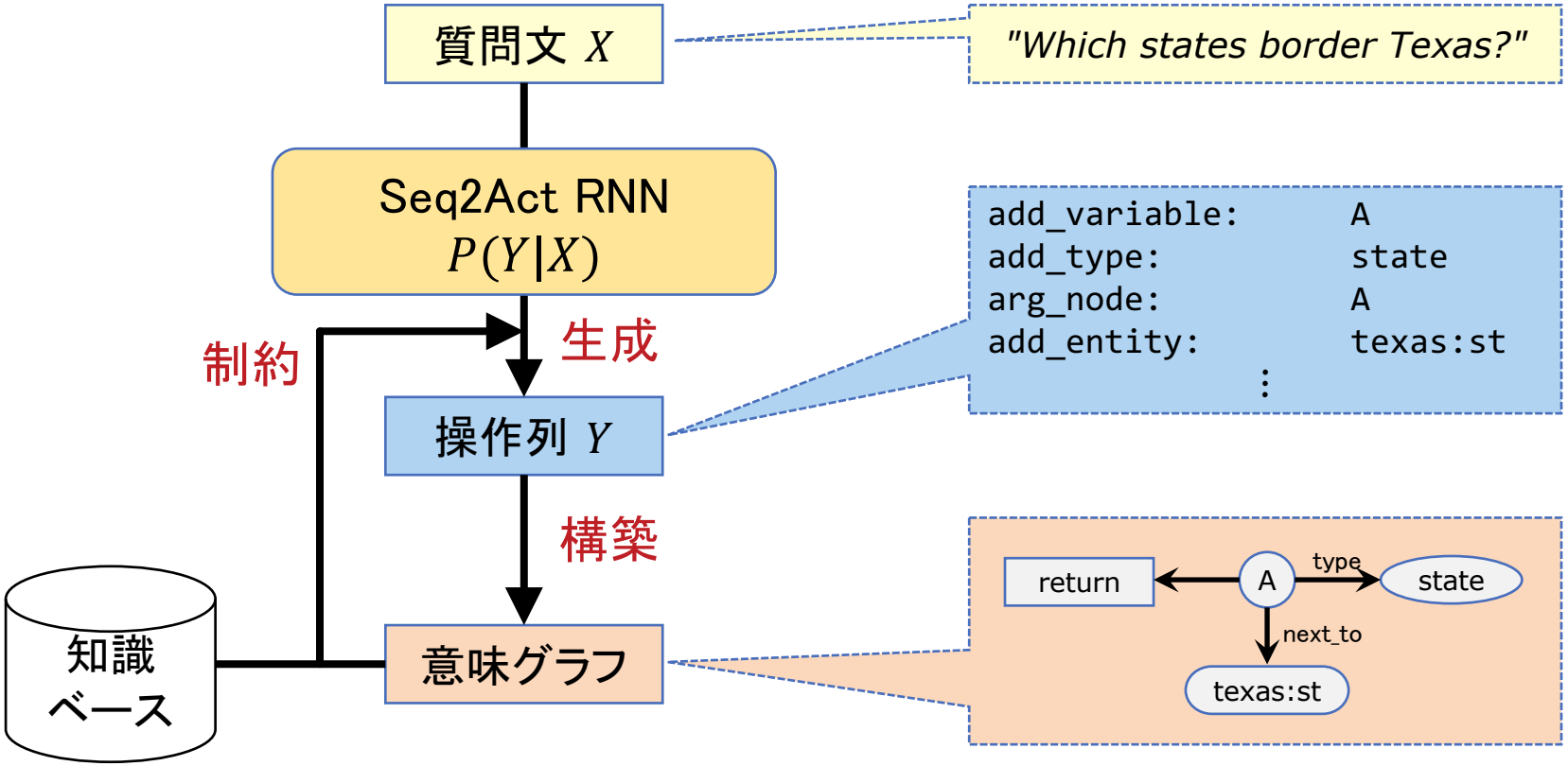
"Which states border Texas?"
に対応する意味グラフ

3. Seq2Seq RNN モデル に基づく手法 [Xiao et al., 2016]他
   - 「質問文→論理式」の翻訳を学習，end-to-end，最近の流行り
   - linearize された論理式から構造的・意味的制約を抽出することは困難

論文の概要

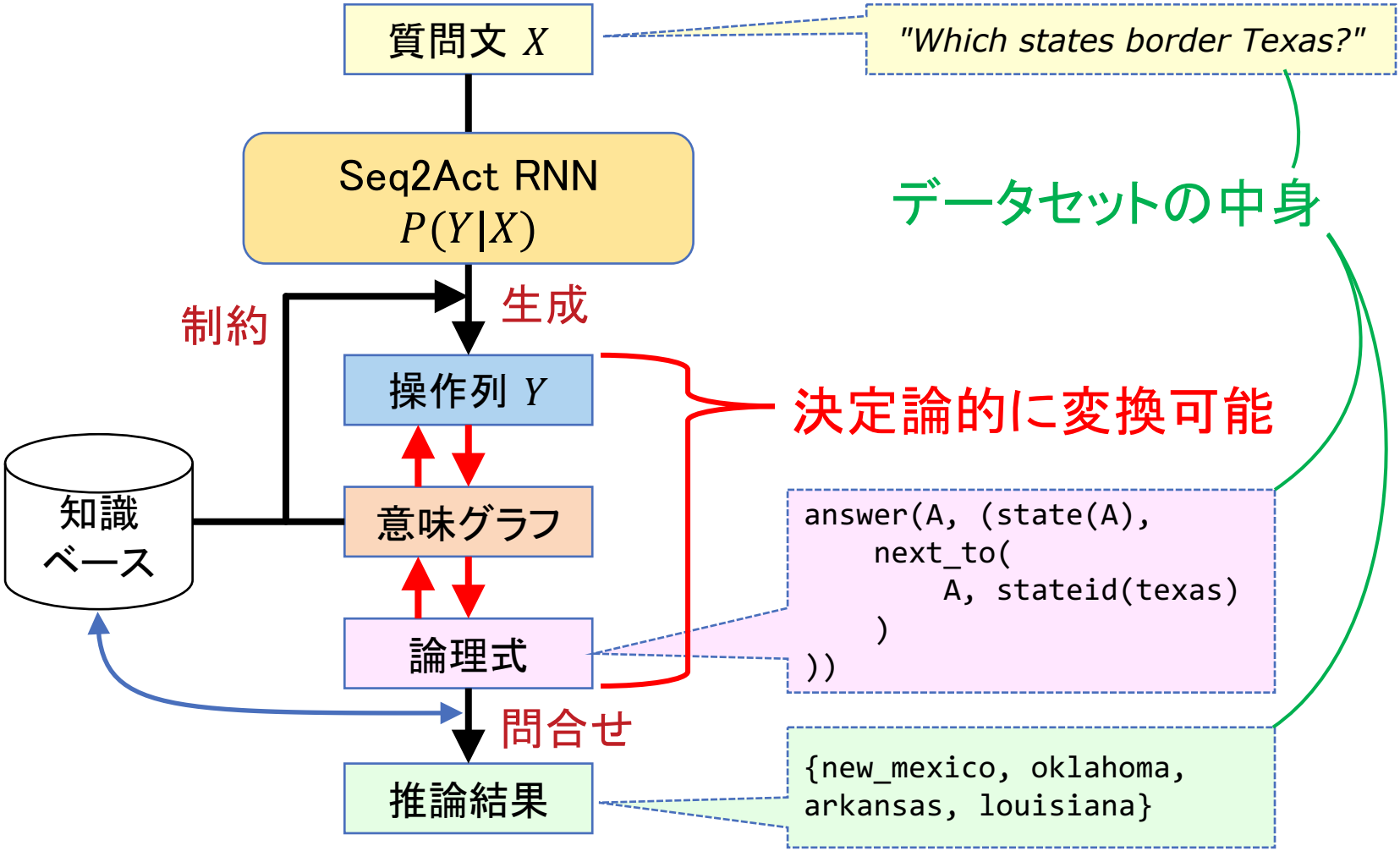
- 意味解析のためのSeq2Seqモデル<u>Seq2Act</u>を提案
  - 入力: 質問文 $X = x_1, \cdots, x_m$
  - 出力: X に対応する意味グラフを構築するための操作列 (action sequence) $Y = y_1, \cdots, y_n$
- 前述の2つの手法のいいとこ取り
  - 意味グラフ → グラフとしての整合性から候補を絞れる
知識ベースの情報も利用可能
  - Seq2Seq → 強力な学習・予測能力を利用可能
語彙項目・文法等を人手で作らずに済む
end-to-end で学習できる
- 3種類のデータセットで評価，他手法と同等以上
  - グラフとしての整合性からくる制約も性能向上に寄与

提案手法の概略 (1/2)



論文 Figure 1 の例

提案手法の概略 (2/2)


質問文 X
"Which states border Texas?"
Seq2Act RNN
$P(Y|X)$
データセットの中身
制約
生成
操作列 Y
決定論的に変換可能
知識
ベース
意味グラフ
answer(A, (state(A),
next_to(
A, stateid(texas)
)
))
論理式
問合せ
推論結果
{new_mexico, oklahoma,
arkansas, louisiana}

意味グラフを構築する操作の設計

- 次の6種類の操作を定義
  - これらの操作のシーケンスから意味グラフを構築する

操作の種類	内容	操作の例
変数ノードの追加	変数ノード (A, B 等) をグラフに追加する. 戻り値あるいは中間変数を表す.	`add_variable:A` `add_variable:B`
エンティティノードの追加	エンティティノード (Texas, New York 等) をグラフに追加する.	`add_entity_node:texas` `add_entity_node:1000`
型ノードの追加	型ノード (`state`, `city` 等) をグラフに追加する.	`add_type_node:state` `add_type_node:river`
エッジの追加	2つのノードの間にエッジを追加する.	`add_edge:next_to` `add_edge:traverse`
演算操作	最大・最小などの演算のスコープを指定する. `start` から `end` までがスコープ.	`start_operation:most` `end_operation:most`
引数の指定	エッジがどのノードを繋ぐか指定する. 型ノード・エッジの追加や演算と一緒に使う.	`arg:A` `arg1_node:texas`

意味グラフの構築例 (1/7)

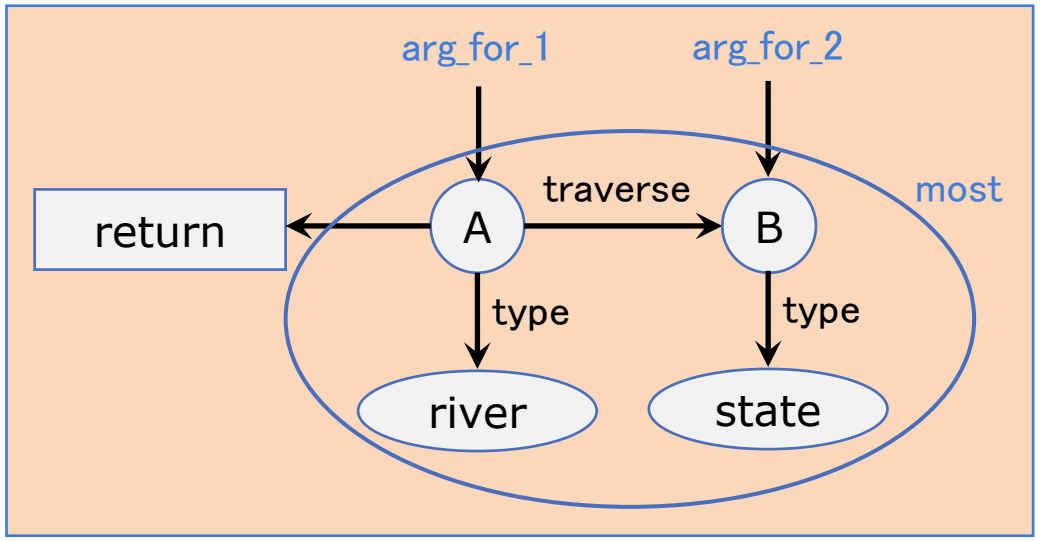
質問文 X

"Which river runs through the most states?"

操作列 Y

structure	Semantic	Arg
add_operation	most	
add_variable	A	
add_type	river	A
add_variable	B	
add_type	state	B
add_edge	traverse	A, B
end_operation	most	A, B
return	A	

意味グラフ



意味グラフの構築例 (2/7)

質問文 X

"Which river runs through the most states?"

操作列 Y

structure	Semantic	Arg
add_operation	most	
add_variable	A	
add_type	river	A
add_variable	B	
add_type	state	B
add_edge	traverse	A, B
end_operation	most	A, B
return	A	

実際には
add_type_node:river
arg:A
の2つだが，表では1つにまとめている

add_edge:traverse
arg1_node:A
arg2_node:B

end_operation:most
arg_for_1:A
arg_for_2:B

意味グラフの構築例 (3/7)

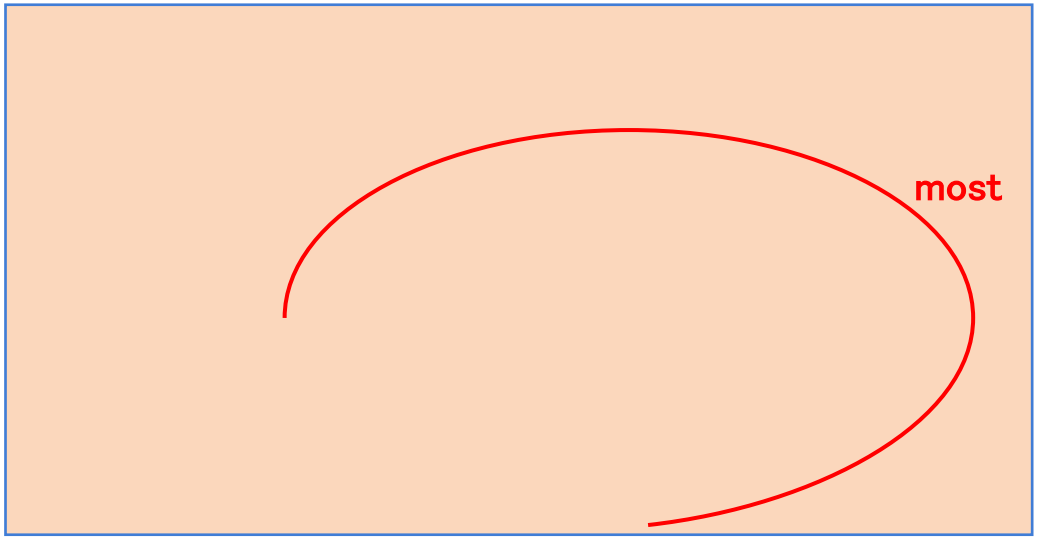
質問文 X

"Which river runs through the most states?"

操作列 Y

structure	Semantic	Arg
add_operation	most	
add_variable	A	
add_type	river	A
add_variable	B	
add_type	state	B
add_edge	traverse	A, B
end_operation	most	A, B
return	A	

意味グラフ



意味グラフの構築例 (4/7)

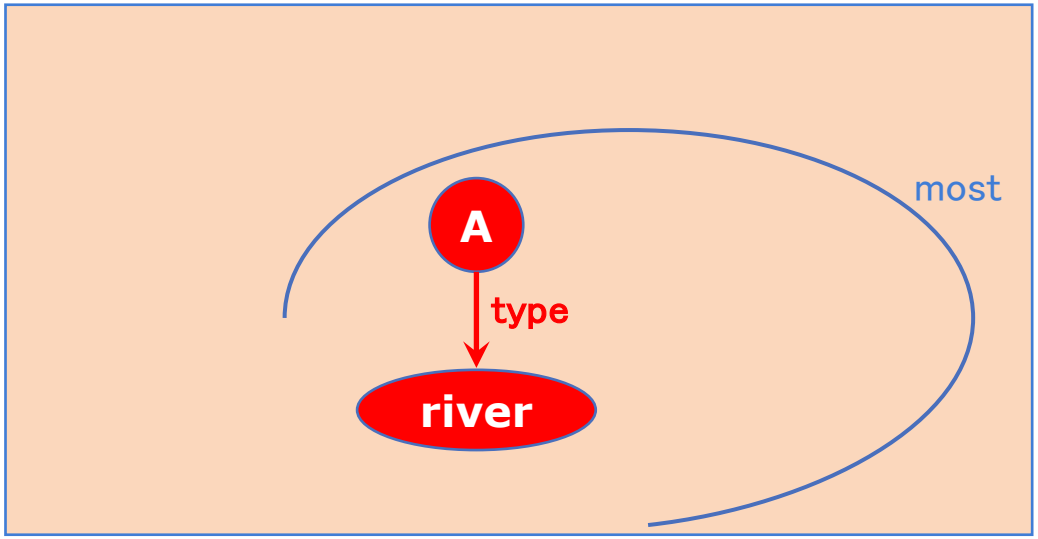
質問文 X

"Which river runs through the most states?"

操作列 Y

structure	Semantic	Arg
add_operation	most	
add_variable	A	
add_type	river	A
add_variable	B	
add_type	state	B
add_edge	traverse	A, B
end_operation	most	A, B
return	A	

意味グラフ



意味グラフの構築例 (5/7)

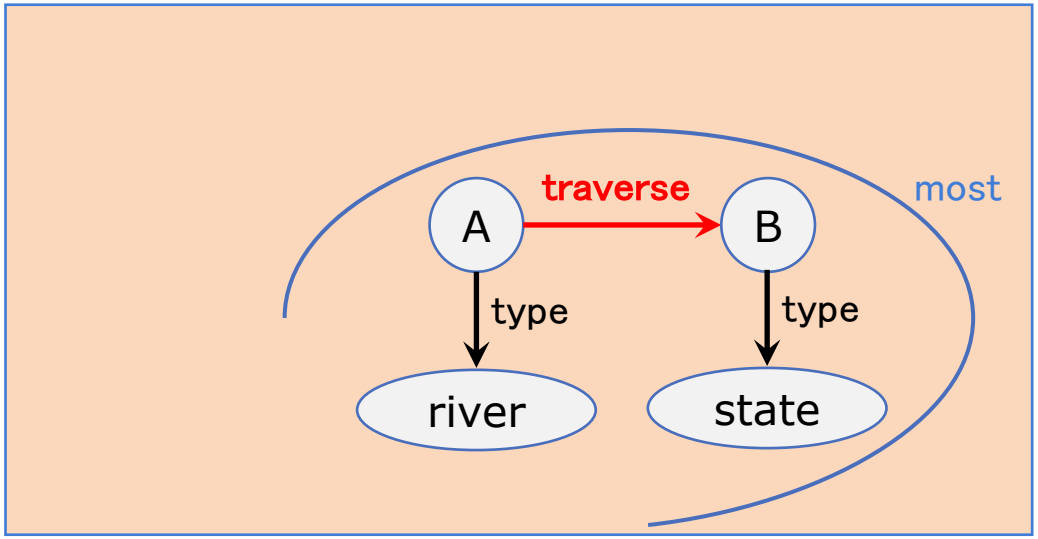
質問文 X

"Which river runs through the most states?"

操作列 Y

structure	Semantic	Arg
add_operation	most	
add_variable	A	
add_type	river	A
add_variable	B	
add_type	state	B
add_edge	traverse	A, B
end_operation	most	A, B
return	A	

意味グラフ



意味グラフの構築例 (6/7)

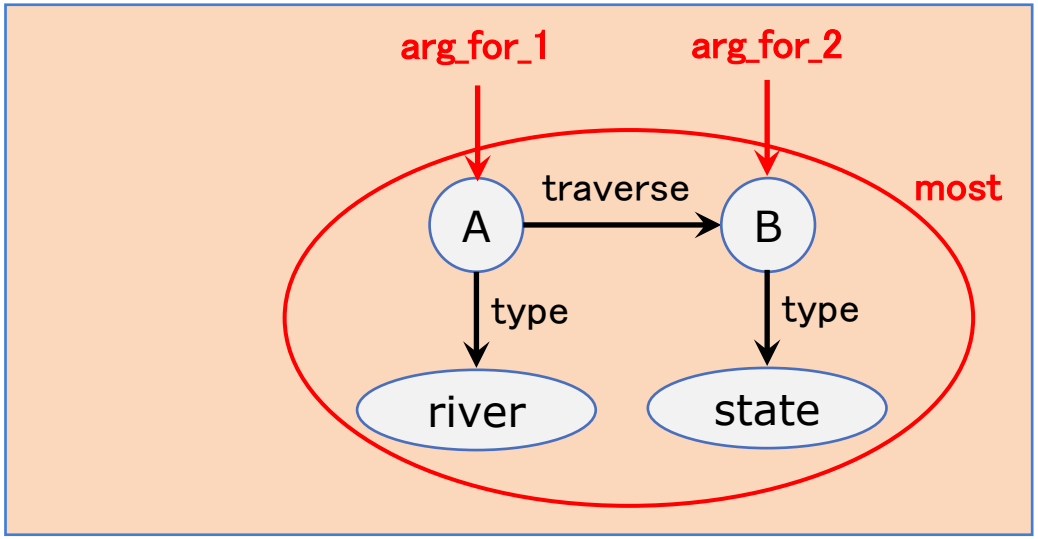
質問文 X

"Which river runs through the most states?"

操作列 Y

structure	Semantic	Arg
add_operation	most	
add_variable	A	
add_type	river	A
add_variable	B	
add_type	state	B
add_edge	traverse	A, B
end_operation	most	A, B
return	A	

意味グラフ



意味グラフの構築例 (7/7)

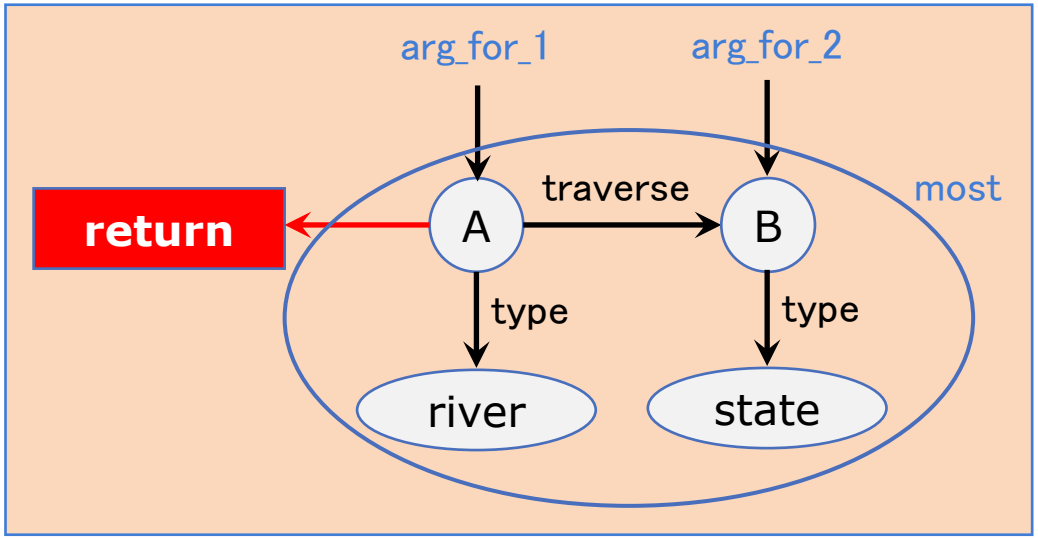
質問文 X

"Which river runs through the most states?"

操作列 Y

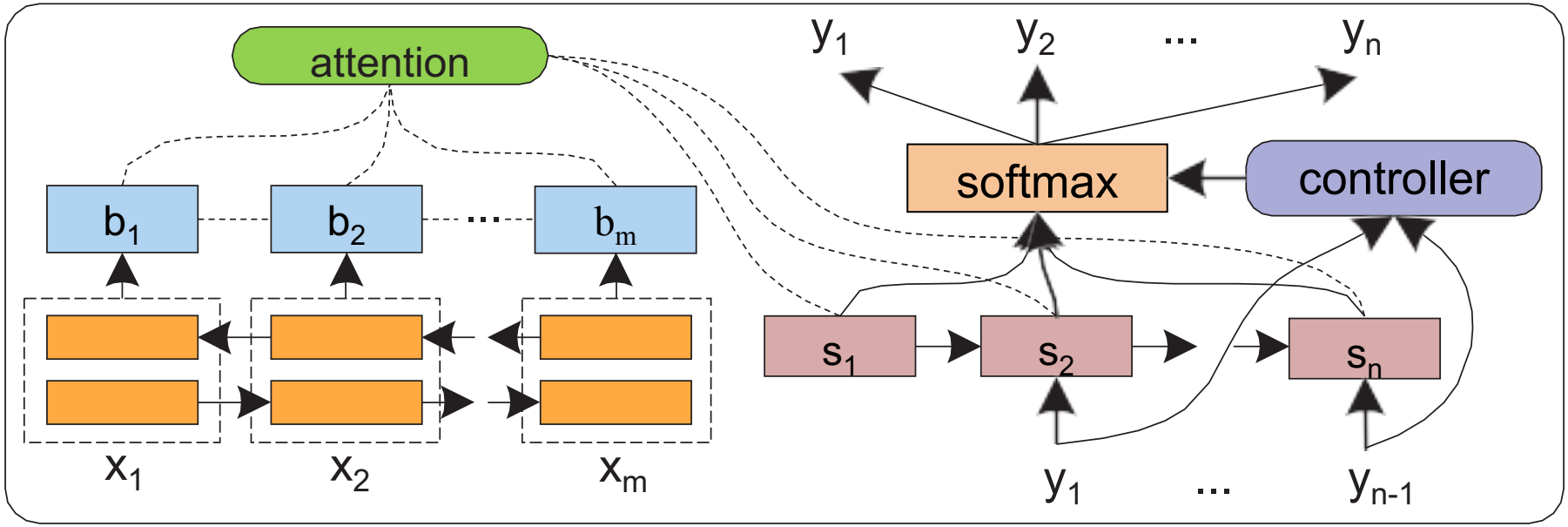
structure	Semantic	Arg
add_operation	most	
add_variable	A	
add_type	river	A
add_variable	B	
add_type	state	B
add_edge	traverse	A, B
end_operation	most	A, B
return	A	

意味グラフ



Seq2Act モデルの設計

- 質問文 X から，意味グラフを構築するための操作列 Y を出力する Seq2Seq モデル
  - controller: 制約を満たさない操作を弾く (後述)



Encoder:
bidirectional RNN

Decoder:
attention-based LSTM

Seq2Act モデルの形式的な定義

- **Encoder**: 入力 $X = x_1, \cdots, x_i, \cdots, x_m$ → 隠れ状態 $b_1, \cdots, b_i, \cdots, b_m$

$$h_i = LSTM\left(\phi^{(x)}(x_i), h_{i-1}\right)$$ 隠れ層ベクトル

$$b_i = \left[h_i^F, h_i^B\right]$$ Forward, Backward を結合

- **Decoder**: 隠れ状態 $b_1, \cdots, b_i, \cdots, b_m$ → 出力 $Y = y_1, \cdots, y_j, \cdots, y_n$

$$s_1 = \tanh\left(W^{(s)}[h_m^F, h_1^B]\right)$$ decoder の state

出力の j 番目を決める際の i 番目の入力トークンに関する attention score（正規化前）

$$e_{ji} = s_j^T W^{(a)} b_i$$

attention weight（正規化後）

$$a_{ji} = \frac{\exp(e_{ji})}{\sum_{i'=1}^{m} \exp(e_{ji'})}$$

context vector（attention weight で加重平均）

$$c_j = \sum_{i=1}^{m} a_{ji} b_i$$

$$P\left(y_j = w | x, y_{1:j-1}\right) \propto \exp\left(U_w[s_j, c_j]\right)$$ state と context を結合，出力を決める

$$s_{j+1} = LSTM\left(\left[\phi^{(y)}(y_j), c_j\right], s_j\right)$$ stateを更新

- 制約を満たさない候補 w は $P = 0$ にする（次のページ）
- 赤字部分を訓練データから学習する

構造的・意味的制約

- 不正なグラフを構築してしまう操作をフィルターする
  - それまでの出力 $y_1, \cdots, y_{j-1}$ から部分的な意味グラフを構築し，y_j の候補が以下の制約を満たすか確認する

1. 構造的制約
   - グラフが非巡回連結グラフとなることを保証する
     (例: next_to エッジは異なるノードを結ばねばならない)
   - 知識ベースに依存しない
2. 意味的制約
   - グラフが知識ベースのスキーマに従うことを保証する
     (例: next_to エッジは2つの state 型ノードを結ばねばならない)
   - 知識ベースに依存する

その他の工夫

- 操作の埋込表現 $\phi^{(y)}$ ：
  - 構造パート $\phi^{(y)}_{struct}$ ・意味パート $\phi^{(y)}_{sem}$ に分ける
  - $\phi^{(y)}(\text{add\_edge:next\_to}) = \left[\phi^{(y)}_{struct}(\text{add\_edge}), \phi^{(y)}_{sem}(\text{next\_to})\right]$
- エンティティを扱うための仕組み：次の2つを試す
  1. 型付きユニークIDで置き換える [Dong et al., 2016]
     - *"jobs with a salary of 40000"* → *"jobs with a salary of* $\underline{num_0}$*"*
     - pre-processing + post-processing
  2. attention-based copying で生成 [Jia et al., 2016]
     - 入力文のトークンを出力へそのままコピーする仕組み
     - ボキャブラリ+入力内のトークン から確率の高いものを出力する

評価手法

- Datasets: GEO, ATIS, OVERNIGHT の3種類
  (自然言語文の質問とそれに対応するクエリを含む)
- 比較手法: 既存の複数の意味解析手法
  制約を用いない Seq2Act も用意
- 実験設定: [Jia et al., 2016] に従う
  (隠れユニット数200, 単語の埋込表現100次元, …)
- 評価指標: 論理式を知識ベースに問い合わせて
  得られた推論結果の accuracy
  ("standard accuracy metrics")

実験結果 (GEO, ATIS)

- 先行研究に匹敵する性能，太字の2つにのみ劣る

	GEO	ATIS
Previous Work		
Zettlemoyer and Collins (2005)	79.3	–
Zettlemoyer and Collins (2007)	86.1	84.6
Kwiatkowksi et al. (2010)	88.9	–
Kwiatkowski et al. (2011)	88.6	82.8
Liang et al. (2011)* (+lexicon)	**91.1**	–
Poon (2013)	–	83.5
Zhao et al. (2015)	88.9	84.2
Rabinovich et al. (2017)	87.1	**85.9**
Seq2Seq Models		
Jia and Liang (2016)	85.0	76.3
Jia and Liang (2016)* (+data)	89.3	83.3
Dong and Lapata (2016): 2Seq	84.6	84.2
Dong and Lapata (2016): 2Tree	87.1	84.6
Our Models		
Seq2Act	87.5	84.6
Seq2Act (+C1)	88.2	85.0
Seq2Act (+C1+C2)	88.9	85.5

性能が高い既存手法2つ:
人手で作った辞書を用いる等の工夫が入っている

Seq2Seq 型では提案モデルが最良
↓
操作列 encoding の有効性を示す
(論理式を直接予測するのに比べて)

構造的制約(C1),
意味的制約(C2) の効果を確認

実験結果 (OVERNIGHT)

- 提案手法は，どのドメインに対しても最良
  - 提案手法の汎用性を示す
  - 構造的・意味的制約は OVERNIGHT でも有効に機能

	Social	Blocks	Basketball	Restaurants	Calendar	Housing	Publications	Recipes	
	Soc.	Blo.	Bas.	Res.	Cal.	Hou.	Pub.	Rec.	Avg.
Previous Work									
Wang et al. (2015b)	48.2	41.9	46.3	75.9	74.4	54.0	59.0	70.8	58.8
Seq2Seq Models									
Xiao et al. (2016)	80.0	55.6	80.5	80.1	75.0	61.9	75.8	–	72.7
Jia and Liang (2016)	81.4	58.1	85.2	76.2	78.0	71.4	76.4	79.6	75.8
Jia and Liang (2016)* (+data)	79.6	60.2	87.5	79.5	81.0	72.5	78.3	81.0	77.5
Our Models									
Seq2Act	81.4	60.4	87.5	79.8	81.0	73.0	79.5	81.5	78.0
Seq2Act (+C1)	81.8	60.9	88.0	80.1	81.0	73.5	80.1	82.0	78.4
Seq2Act (+C1+C2)	**82.1**	**61.4**	**88.2**	**80.7**	**81.5**	**74.1**	**80.7**	**82.9**	**79.0**

その他の分析

- エンティティの表現
  - 型付きユニークIDによる置換が優位

	Replacing		*Copying*
GEO	88.9	>	88.2
ATIS	85.5	>	84.0
OVERNIGHT	79.0	>	77.9

Table 3: Accuracy の比較
(*Replacing*: 型付きユニーク ID による置換)
(*Copying*: Attention-based copying)

- 操作列 encoding がなぜ有効か
  - 系列長は平均的に (論理式) > (操作列) になっている
  - 長距離依存性の問題の影響を低減できる

	Logical Form		Action Sequence
GEO	28.2	>	18.2
ATIS	28.4	>	25.8
OVERNIGHT	46.6	>	33.3

Table 4: 平均系列長の比較

エラー分析

- 主に2種類のエラーがあることがわかった

1. 出現が稀/インフォーマルな構文を解釈できない
   - 例: *"Iowa borders how many states?"*
     - *"How many states does Iowa border?"* がフォーマルな構文
   - フォーマルな統語構造になるように前処理する必要あり
2. Under-Mapping (論理式要素の生成漏れ)
   - 入力質問文の一部トークンが出力時に無視される現象
   - ニューラル機械翻訳の訳抜けと同じ問題

まとめ

- 対象タスク: 意味解析 (Semantic Parsing)
- 従来手法: 意味グラフ … ヒューリスティクス頼り
  Seq2Seq … linearize された論理式からの制約抽出は困難
- 提案手法: 意味グラフ構築操作列をSeq2Seq予測
  decoding時にグラフ由来の制約を活用
- 分析結果: 先行研究を上回るか同等程度の性能
  論理式に比べて系列長を短くできる
  グラフ由来の制約は性能向上に寄与