強化学習によるプログラム生成のためのプログラム系列分析

佐藤 拓海
東北大学 工学部 電気情報物理工学科

1 はじめに

表形式データに基づく質問応答 (TableQA) は，一つの表とそれに関連する質問が与えられ，適切な応答を返すタスクである．図1はその例を表している．一つの表とそれに関する質問 “Who is ranked #1?” が与えられ，“Cuba” と答えられれば正解となる．正解応答を導くには，表に対する何らかの操作が必要となる．そのため，SQL のクエリのように表の操作を実行可能なプログラムを出力するのが一般的である．出力されたプログラムはインタプリタによって実行され，表に基づいて値を返す．図1中の P_1 がその例である．まず，“(first all_rows)” によって，全ての行のうち最初の行を取り出す．次に，“(hop v0 r.nation-str)” によって，取り出した行の “Nation” 列の文字列 “Cuba” を取り出す．

TableQA に用いられるデータセットでは，各質問に対する正解応答のアノテーションのみが与えられ，正解応答を導くプログラム (正解プログラム) のアノテーションはないことが多い [4, 6, 7]．その理由として，質問と応答のペアは人手でのアノテーションコストが低いが，正解プログラムのアノテーションはコストが高いことが挙げられる．最先端の手法では強化学習を用いて，正解応答にたどり着くプログラムに高い報酬を設定し，繰り返し探索することによって尤もらしいプログラムを導出する [2]．

ここで問題となるのは，正解プログラムのアノテーションがないため，モデルが生成したプログラムの分析が困難な点である．実際に多くの先行研究 [4, 8] では，最終的に出力した応答の正解率のみでモデルを評価するに留まり，誤った応答を出力するに至ったプログラムの分析などが行われていない．その結果，モデルの挙動に関する詳細が未だ明らかではない．

そこで本稿では，(i) 質問と応答のアノテーションのみから正解プログラムを獲得する方法と，(ii) それに基づく詳細な評価・分析が可能なフレームワークを提案する．本提案手法を用いることにより，モデルが出力したプログラム内の個々の関数の正解率など詳細な分析が可能であることを示す．

Rank	Nation	Gold	Silver	Bronze	Total
1	Cuba	4	3	2	9
2	Canada	4	2	1	7
3	United States	2	0	2	4
4	Mexico	1	1	0	2
...	...	...	...	...	...

Q: “Who is ranked #1?”
A: “Cuba”
P_1: (first all_rows)
(hop v0 r.nation-str)
P_2: (filter_{in} all_rows [“1”] r.rank-str)
(hop v0 r.nation-str)

図1: WikiTableQuestions データセットの例．表と，質問 (Q)，正解の応答 (A)，および本提案手法で生成した正解プログラム群 (P)．

2 関連研究

これまでもいくつかの研究において，モデルの出力したプログラムに関する分析が行われている．Pasupat ら [6] は，TableQA の WikiTableQuestions データセット内の質問を四つに分類し，誤りの割合を算出している．この研究は正解の応答に基づいたものであり，出力されたプログラムがどの程度誤っていたのかなどの詳細な分析には踏み込んでいない．Krishnamurthy ら [1] は同様のデータセットを用いて，seq2seq モデルの出力するプログラムに関する分析を行っている．開発データにおいて正解の応答を出力できなかったプログラム 100 件について，生成候補を見てエラーの原因の分類を行なっている．この研究のように，正解プログラムのアノテーションがないため，出力された少数のプログラムを人手で一つ一つ見ていくといったコストの高い方法が取られている．本稿で提案する分析手法では，正解プログラムを自動で獲得するため，出力されたプログラムの分析を大幅に効率化できる．

3 提案手法

本節では，プログラムの評価・分析手法を提案する．図2にその概要を示す．ここでは，①正解プログラムの生成，②実験，③評価の 3 つのステップを踏む．以降の

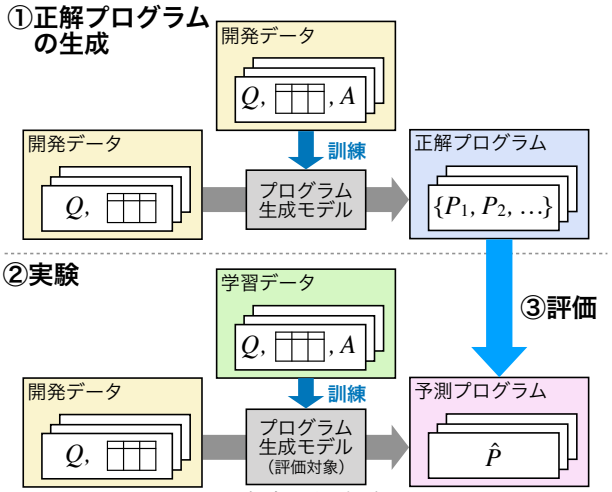

図2: 提案手法の概略図.

節に詳述する.

3.1 分析対象のタスク

TableQA タスクを分析対象とする (図1). このタスクでは，与えられた表に関する質問 (Q) に対し，適切な応答 (A) を出力することがもとめられる．その解法として，質問と表を入力とし，SQL のクエリなどの表上で実行可能なプログラムを出力するモデルを構築し，モデルの出力であるプログラムを表上で実行することで解答を得るのが一般的である．本タスクで用いられるデータセットの多くは，各質問に対する正解の応答のみがアノテーションされており，正解プログラムのアノテーションは存在しない．本提案手法はそのようなデータセットに対し，正解プログラムを自動獲得することを目的とする．

3.2 正解プログラムの生成法

モデルが予測したプログラムを分析するにあたり，比較対象となる正解プログラムを用意する必要がある．

正解プログラムの定義: 各質問に対して正解の応答を導くプログラムを「正解プログラム」と定義する．

正解プログラムの生成方法: 図2の①に概略を示す．まず，正解プログラムを付与したい質問・応答アノテーション付きデータを用いて，正解の応答を予測できるようにモデルを学習する．次に，この学習済みモデルを用いて，学習に用いたデータ内の各質問に対するプログラムを出力する．ここで，学習に用いたデータとプログラムを出力するデータは同一のデータである．つまり，学習に用いたデータに対してプログラムの予測を行うため，多くの質問に対して正解の応答が予測できるので正解プログラムも容易に獲得できる．

正解プログラムの曖昧性: 前述した手法で，多くの質問に対する正解プログラムが獲得できる．ここで一つ問題となるのは，図1の P_1 と P_2 のように，正解プログラムは複数ありうるということである．例えば，データ内の各質問・応答に対して，正解プログラムを一つ用意する場合を考える．モデルが実際に生成したプログラムを評価する際，本当は正解であるはずが，用意した正解プログラムと異なるため不正解としてしまうという問題を引き起こしうる．

このような曖昧性に対処するため，本稿では，複数の異なる学習済みモデルを用意し，複数の正解プログラムを獲得する．各モデルはニューラルネットワークを用いているため，パラメータのランダム初期化の際に，ランダムシード値が変われば，それに伴って異なるモデルが構築される．この性質を利用し，ランダムシードの異なる複数のモデルを用意し，それぞれ独立に学習する．これによって，複数の異なる正解プログラムが獲得可能であり，曖昧性が引き起こす問題を軽減できる (実験結果は5.1節を参照).

3.3 正解プログラムの評価法

正解プログラムとモデルの予測プログラムを比較することによって，予測プログラムがどの程度適切なものか評価する (図2の③の出力). 具体的には，(i)BLEU スコア [5] を用いてプログラム全体の評価をする．さらに詳細な分析のため，(ii) プログラム内の関数の N-gram 一致率を算出し，各関数別の評価も行う.

(i) プログラム全体の評価: BLEU スコアの算出では，まず，予測プログラム $\hat{P}$ とそれぞれの正解プログラム $\{P_1, P_2, \cdots\}$ 間の BLEU スコアを算出する．これらの中で，BLEU スコアが最大となる正解プログラムを「最類似正解プログラム」とする．最類似正解プログラムの BLEU スコアを全 QA 対について平均した値を最終スコアとする.

(ii) 関数ごとの評価: 予測プログラムに含まれる各関数を評価する．まず，予測プログラムと最類似正解プログラムから，引数を無視し関数名のみの系列を作る．引数を無視する理由は，プログラム内で中心的な役割を果たす関数に焦点を絞った量的評価を行いやすくするためである．次に，予測プログラムの関数名系列の N-gram が，最類似正解プログラムの関数名系列に含まれているかどうかに基づいて F_1 値を算出する.

4 実験

本提案手法の有用性を検証するための実験を行う．本節では，実験に用いるデータセットとモデルについて詳述する.

4.1 データセット

WikiTableQuestions データセット [6] を用いる．このデータセットの質問と応答は人手で作成されており，人間が実際に行う質問応答により近いものとなっている．このデータセットには，QA 対が 22,033 個，表が 2,108 個存在する．QA 対のうち 11,321 個を学習データ，2,831 個を開発データとして用いており，両データ間で用いられる表は互いに異なるものである．一方で，正解プログラムはアノテーションされていない．そこで本提案手法を用いて，開発データに正解プログラムを付与する．そして，図2中②に示すように，学習データで学習した評価対象のモデルが予測したプログラムに対して評価・分析を行う．

4.2 分析に用いるモデル

本提案手法は任意のモデルを評価・分析可能である．本稿では,WikiTableQuestions データセットにおいて最高精度を達成しているモデル [2] を用いる.このモデルは，強化学習の枠組みである Memory Augmented Policy Optimization (MAPO) を Neural Symbolic Machines (NSM)[3] モデルに適用したものである．

MAPO は方策勾配法 (policy gradient algorithm) に基づいて，モデルのパラメータを学習する．高報酬な行動軌跡をメモリバッファに保存し，期待報酬をメモリバッファ内外の各行動軌跡の重み付き和とすることで，高報酬な行動軌跡を記憶・利用することが特徴である．NSM は質問と表の情報を入力とし，表上で実行可能なプログラムを出力する seq2seq と，入力されたプログラムを表上で実行し，応答を出力するインタープリタを合わせたモデルである．その特徴は，seq2seq のデコード時に質問や表のエンティティ，すでに生成され実行可能な関数の実行結果をメモリに保持しておくことで，それらを利用しつつプログラムを生成できる点である．

このモデルが生成するプログラムは図1に示すように，丸括弧で区切られる関数の系列となっており，各関数は関数名と引数からなるトークンの系列となっている．ハイパーパラメータの設定は [3] と同様のものを用いる．学習ステップ (training step) 数の上限は，正解プログラムの生成を行うモデルに対しては 10 万回，実験に用いるモデルには 3 万回と設定した．

表1: モデル数を変化させた場合の正解プログラムとの一致数と正解応答をカバーできていない正解プログラム数．

	モデル数				
	1	2	3	4	5
`total(t)`	2,186	2,221	2,235	2,241	2,253
`eq(t,p)`	836	951	1,011	1,067	1,093
`neq(t,p)`	1,350	1,270	1,224	1,174	1,160
`error(t,p)`	358	243	183	127	101

5 結果・考察

5.1 正解プログラムの正解カバー率

3節で述べたように，正解の応答を出力可能なプログラムは複数存在する．そのため，モデルが予測したプログラムが本当は正解であるはずが，用意された正解プログラムと一致しないため，誤った正解・不正解判定をしてしまう恐れがある．そこで，本提案手法によって獲得された正解プログラムが，実際にモデルが出力した予測プログラムをどの程度適切に正解判定できているかを調査する．

表1に実験結果を示す．`total(t)` は，最低一つの正解プログラム `t` を獲得できた QA 対の総数 `total(t)` を表す．`eq(t,p)` は，最低一つの正解プログラム `t` と完全一致した予測プログラム `p` の総数を表す．`neq(t,p)` は，全正解プログラム `t` と不一致である予測プログラムの数を表す．`error(t,p)` は，本来正解と判定されるべきであった予測プログラムの総数を表す．

一つのモデルの正解プログラムと予測プログラムを完全一致基準で判定した際，予測プログラムの 358 個 (2,186 中) は本来正解と判定されるべきだが正解プログラムと一致しないため不正解と判定されている．このような誤った判定数は，モデルの数を増やして正解プログラムの数を増やすと徐々に減っていく．五つのモデルによる正解プログラムを使った場合，この数は 101 個 (全体の約 4.5%) にまで減少する．以上の結果から，提案した正解プログラムの出力方法によって複数の正解プログラムを用意することによって，モデルの予測したプログラムのほとんど (95% 以上) を正確に正解・不正解判定できるということがわかった．

5.2 予測プログラムの評価・分析

3.3節で述べた評価方法に基づいて，学習済みの最先端のモデルが出力したプログラムを評価・分析する．

プログラム全体に対する評価： 各予測プログラムの BLEU 値を算出することによって，各プログラムがどの程度正解プログラムと異なっていたかがわかる．結果として，予測プログラムの BLEU スコアの平均値は 77.0

であった．この中でも，BLEU スコアが 20 を下回っている予測プログラムとそれに対応する QA 対について分析したところ，質問文中と表中の単語間に表記ゆれがある事例が多いことがわかった．例えば，質問文 “how many of the candidates were females?” を考える．この質問文に付随する表中には “female” を “F” と略して表している．このように，質問文中の単語とそれに対応する表中の単語の表記ゆれが，プログラムの予測を困難なものとしていることが示唆された．

関数ごとの評価： 提案手法を用いることによって，各関数の精度が詳細に分析可能である．表2は，正解プログラム中に含まれている各関数の事例数と F_1 値を示している．unigram の列は，各関数単体の事例数と F_1 値を示している．全体の傾向として，事例数の多い関数 (hop や filter$_{in}$，count) の F_1 値は高く，予測しやすいことがわかる．bigram の列は，当該関数とその直前の関数のペアに対する事例数と F_1 値を示している．つまり，直前に出力される関数とセットでの精度である．結果として，関数 previous や next は，unigram と bigram での F_1 値がほぼ同等である．これは，直前の関数とセットで正解できていることを示唆している．一方，関数 filter$_{in}$ は，unigram での F_1 値よりも bigram の F_1 値が顕著に低い．これは，正解の応答を導くためには関数 filter$_{in}$ が必要であることをモデルが正しく認識してはいるが，直前に用いる関数として適切な関数を選べていないことを表している．

表2: 関数別の事例数と F_1 値．

	unigram		bigram	
	事例数	F_1	事例数	F_1
hop	1,528	92.2	1,445	70.1
argmax	340	68.0	101	56.8
argmin	203	62.4	12	0.0
filter$_{>}$	48	63.2	3	0.0
filter$_{\geq}$	61	53.1	13	0.0
filter$_{<}$	27	57.1	1	0.0
filter$_{\leq}$	0	0.0	0	0.0
filter$_{=}$	67	63.0	11	50.0
filter$_{\neq}$	14	0.0	9	0.0
filter$_{in}$	1,269	91.7	150	64.1
filter$_{!in}$	38	66.7	23	60.0
first	202	63.2	77	43.5
last	87	68.6	31	42.1
previous	93	79.1	93	75.8
next	115	74.2	115	71.7
count	627	83.8	559	65.8
max	24	0.0	10	0.0
min	15	0.0	3	0.0
average	10	42.9	9	16.7
sum	23	32.4	11	13.3
mode	45	51.1	4	0.0
same_as	34	76.9	34	76.9
diff	78	77.7	78	77.7

6 おわりに

本稿では，TableQA において生成されたプログラムの評価・分析フレームワークを提案した．実験の結果，本提案手法の妥当性の評価と，実際に最先端の TableQA モデルに適用した際の評価・分析を行った．今後の課題として，より幅広いデータセットとモデルに対して本提案手法を適用し，評価・分析を行うことが挙げられる．

謝辞

本研究の一部は JST CREST（JPMJCR1301）および日本電信電話株式会社の支援を受けて行った．

参考文献

[1] Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. “Neural Semantic Parsing with Type Constraints for Semi-Structured Tables”. In: *Proceedings of EMNLP*. 2017, pp. 1516–1526.

[2] Chen Liang et al. “Memory Augmented Policy Optimization for Program Synthesis and Semantic Parsing”. In: *Proceedings of NIPS*. 2018, pp. 10014–10026.

[3] Chen Liang et al. “Neural Symbolic Machines: Learning Semantic Parsers on Freebase with Weak Supervision”. In: *Proceedings of ACL*. Vol. 1. 2017, pp. 23–33.

[4] Arvind Neelakantan, Quoc V. Le, and Ilya Sutskever. “Neural Programmer: Inducing Latent Programs with Gradient Descent”. In: *CoRR* abs/1511.04834 (2015).

[5] Kishore Papineni et al. “Bleu: a Method for Automatic Evaluation of Machine Translation”. In: *Proceedings of ACL*. 2002.

[6] Panupong Pasupat and Percy Liang. “Compositional Semantic Parsing on Semi-Structured Tables”. In: *Proceedings of ACL-IJCNLP*. 2015, pp. 1470–1480.

[7] Pengcheng Yin et al. “Neural Enquirer: Learning to Query Tables in Natural Language”. In: *Proceedings of the Workshop on Human-Computer Question Answering*. 2016, pp. 29–35.

[8] Yuchen Zhang, Panupong Pasupat, and Percy Liang. “Macro Grammars and Holistic Triggering for Efficient Semantic Parsing”. In: *Proceedings of EMNLP*. 2017, pp. 1214–1223.