事前学習済み Transformer を用いた Data-to-text における入力順序の影響分析

矢野 祐貴　　須藤 克仁　　中村 哲
奈良先端科学技術大学院大学
{yano.yuki.yt0, sudoh, s-nakamura}@is.naist.jp

1 はじめに

Data-to-text は構造化データを入力として非構造化データである自然言語文を生成するタスクである [1]．具体的に，構造化データは表や知識グラフなどの様々な形式で与えられ，出力はこれらの入力を説明あるいは要約する自然言語文である．また，構造化データはいずれの形式においてもレコードの集合として定式化する事ができる．

近年では，ニューラルネットワークを用いたモデルが高い性能を発揮しており，特に，知識グラフを入力形式とするデータセットにおいては，Transformer [2] の事前学習済みモデルである T5 [3] を用いた研究が state-of-the-art の精度を達成し，強力なベースラインとなっている [4, 5]．一方で，スコアボードの表のような複雑な表形式の構造化データを入力とするデータセットにおいては，レコードの順序やそのコンテンツのプランニングおよび選択が生成の精度に大きく寄与することが示され，注目を集めている [6]．

これまでの知識グラフを入力形式とする T5 を用いた研究では，知識グラフを線形化によってコンテンツの系列として入力する．ここで構造化データを構成するレコードの順序は考慮されておらず，文献 [6] のような順序のプランニングも行われず，頒布されたデータセットにおける順序がそのまま利用されている．そこで本研究では，T5 を用いた知識グラフを扱う Data-to-text モデルに対し，入力レコードの順序が生成文に及ぼす影響を調査する．

実験では，Data-to-text で標準的に用いられる知識グラフとその説明文のデータセットである WebNLG [7] を用いて，T5 を用いた最初の Data-to-text モデル [4] に対し，学習用およびテスト用データのレコードを「そのままの順序」／「シャッフルした順序」の 2 パターンで Fine-tuning および生成を行う．それぞれのパターンの生成結果を BLEU [8] などの自動評価指標で評価し，レコード順序による生成文への影響を分析する．また，生成結果の入力レコード数ごとの BLEU 値を比較し，その傾向について考察する．評価実験の結果，レコード順序をそのままで Fine-tuning したモデルに，レコード順序をシャッフルしたテストデータで生成すると，そのままの順序で生成した結果よりも，BLEU 値が低下することが分かった．対して，レコードの順序をシャッフルして Fine-tuning すると，既知ドメインのテストデータの BLEU 値が向上することが分かった．また，レコード数ごとの解析では，順序のシャッフルによって，少ないレコード数の入力に対して，BLEU 値が向上することが分かった．これらの評価実験の結果を通して，T5 を用いたベースラインに対しても，入力順序のプランニングが必要であることが分かった．

2 関連研究

近年複数の Data-to-text のデータセットで，Text-to-text transfer Transformer (T5) が state-of-the-art の精度を達成している [4, 5]．T5 自体は，言語生成タスク（機械翻訳や自動要約，質問応答生成など）を Text-to-text のフレームワークで解く Transformer [2] を用いたアーキテクチャであり，Colossal Clean Crawled Corpus (C4) で事前学習したモデルを下流タスクである各言語生成タスクに適応することで高い精度を示している．

文献 [4] は，Data-to-text を T5 の下流タスクとして解くための手法を調査している．構造化データを線形化しコンテンツの系列とした入力で Fine-tuning するだけで，従来の語彙化やその解除，レコードごとの素性の設計などの作業を必要としない，完全な End-to-end で Data-to-text を解くことができる．結果として，WebNLG, MultiWoz, ToTTo の 3 つのデータ

This work is licensed by the author(s) under CC BY 4.0
(https://creativecommons.org/licenses/by/4.0/).

	主語	述語	目的語
1	Amdavad ni Gufa	location	Gujarat
2	Amdavad ni Gufa	location	Ahmedabad
3	Amdavad ni Gufa	country	India
4	India	leaderName	Narendra Modi

↓ シャッフル

	主語	述語	目的語
2	Amdavad ni Gufa	location	Ahmedabad
4	India	leaderName	Narendra Modi
3	Amdavad ni Gufa	country	India
1	Amdavad ni Gufa	location	Gujarat

線形化 → Amdavad ni Gufa location Gujarat Amdavad ni Gufa location Ahmedabad Amdavad ni Gufa country India India leaderName Narendra Modi

線形化 → Amdavad ni Gufa location Ahmedabad India leaderName Narendra Modi Amdavad ni Gufa country India Amdavad ni Gufa location Gujarat

図 1　レコードのシャッフルと線形化

セットで state-of-the-art の精度を達成した．

また，この調査結果を踏まえて文献 [5] では，WebNLG などのグラフを入力形式とするデータに対して，明示的にグラフ構造を与えることでさらなる性能向上が出来ることを示した．具体的には，線形化の際に各コンテンツの先頭に<Head>, <Relation>, <Tail>といったグラフを示すための特殊トークンを挿入して Fine-tuning を行う．

これらの T5 による精度向上が報告されているデータセットの特徴として，参照文が入力レコードのほぼすべての情報について言及している点が挙げられる．一方で，スコアボードの表のような複雑な表データを入力とする場合，参照文で言及されないレコードが多くなるため，生成内容の取捨選択を含むプランニングが有効であるとされている [6]．

本研究は，ほぼすべての入力情報への言及が必要な WebNLG データにおける，レコードの入力順序の影響を明らかにするものである．

3　実験

実験では，事前学習済み Transformer である T5 を用いた Data-to-text モデルに対しても，入力レコードの順序のプランニングが必要であるかを検証する．具体的には，入力の構造化データを図 1 に示すように，レコードを構成するコンテンツはそのままで，順序をシャッフルしてから線形化を行い，T5 モデルへの入力とする．Fine-tuning 時の学習データと，生成時のテストデータに対して，以下の 3 つの組み合わせで実験を行う．

- もとのレコード順序で Fine-tuning したモデルを，もとのレコード順序で生成（def-def）
- もとのレコード順序で Fine-tuning したモデルを，シャッフルしたレコード順序で生成（def-shuf）
- シャッフルしたレコード順序で Fine-tuning したモデルを，シャッフルしたレコード順序で生成（shuf-shuf）

ここで，もとのレコード順序とは，データセットの配布ファイルに記述されているそのままのレコード順序である．多くの場合，その整列について明記されていないが，例えば図 1 の左上のように，主語でソートされていたり，目的語が “India” のレコード次は，主語が “India” のレコードであるなど，人為的な整列が見受けられる．

実験の結果に対して，def-def をベースラインとし，def-shuf と比較することで，モデルがデータセットの順序を学習しているかどうかを検証する．また，shuf-shuf と比較することで，人為的な整列を崩した入力を T5 モデルが学習できるかどうかを検証する．今回の実験では，shuf-shuf の Fine-tuning 時のレコードのシャッフルは，データの読み込み時に一度のみ行い，エポックごとのシャッフルは行わない．

3.1　データセット

実験には WebNLG [7] データセットを用いる．WebNLG は，DBpedia[1] から抽出された Resource Description Framework トリプル（主語，述語，目的語の 3 つ組）の集合で表現される知識グラフと，クラウドソーシングによって収集された対応する知識グラフの説明文から構成されている．学習，開発およびテストデータはそれぞれ，18,102 対，2,268 対，4,928 対に分かれており，さらにテストデータは，学習，開発データに含まれている既知ドメインのデータ（2,495 対）と含まれない未知ドメインのデータ（2,433 対）に分けられている．

1）　DBpedia: https://wiki.dbpedia.org/

This work is licensed by the author(s) under CC BY 4.0
(https://creativecommons.org/licenses/by/4.0/).

表 1　自動評価結果．**太字**は同一事前学習モデルに対するスコアの最大値を示す．

モデル		既知ドメイン					未知ドメイン				
Trainset	Testset	BLEU	NIST	METEOR	ROUGE	CIDEr	BLEU	NIST	METEOR	ROUGE	CIDEr
T5-Small											
Default	Default	62.65	10.70	45.22	73.39	4.21	**40.28**	**8.34**	36.34	**60.12**	**2.77**
Default	Shuffle	62.10	10.64	44.88	72.48	4.20	40.26	8.32	**36.44**	60.05	**2.77**
Shuffle	Shuffle	**63.96**	**10.83**	**45.72**	**74.19**	**4.36**	39.77	8.28	36.00	59.20	2.74
T5-Base											
Default	Default	63.35	10.84	45.78	73.97	4.37	**45.80**	**9.07**	**38.72**	**62.91**	**3.16**
Default	Shuffle	62.56	10.75	45.44	73.17	4.33	44.42	8.93	38.16	61.79	3.09
Shuffle	Shuffle	**64.37**	**10.89**	**45.94**	**74.18**	**4.43**	44.68	8.89	38.08	61.70	3.09
T5-Large											
Default	Default	62.50	10.71	45.22	73.13	4.33	46.21	9.06	**38.54**	63.69	3.13
Default	Shuffle	61.85	10.66	44.93	72.17	4.29	45.88	8.99	38.12	63.15	3.12
Shuffle	Shuffle	**63.37**	**10.76**	**45.37**	**73.79**	**4.38**	**48.08**	**9.15**	38.35	**63.80**	**3.21**

3.2　実験設定

先行研究 [3, 4] に倣い，バッチサイズを 512，学習率を 0.001 として，学習データによる Fine-tuning を行う．最適化には Adafactor [9] を使用する．T5 モデルの語彙は，SentencePiece [10] を用いてテキストを WordPiece トークン [11, 12] へエンコードし，語彙サイズは 3 万 2 千としている．事前学習された T5 モデルのパラメータ数は，60 万（T5-Small），220 万（T5-Base）および 770 万（T5-Large）の 3 種類を使用する．

評価には，開発データに対して BLEU 値が最も高いエポックのモデルパラメータを用いて，テストデータに対する生成を行った結果で比較する．また，学習は開発データに対する BLEU の最大値が 3 回更新されなかったエポックで早期終了する．自動評価指標には，言語生成で用いられる，BLEU, NIST, METEOR, ROUGE-L, CIDEr を使用する．

3.3　実験結果

実験結果を表 1 に示す．まず，既知ドメインの結果について，モデルのすべてのパラメータサイズ，すべての自動評価指標で，shuf-shuf が def-def の結果を上回っている．反対に，モデルのすべてのパラメータサイズ，すべての自動評価指標で，def-shuf が def-def の結果を下回っている．

次に，未知ドメインの結果について，モデルのパラメータサイズが T5-Small および T5-Base の場合，def-def もしくは def-shuf の結果が shuf-shuf の結果を上回っている．一方，モデルのパラメータサイズが T5-Large の場合には，METEOR 以外の評価指標で，shuf-shuf の生成結果が def-def および def-shuf を上回っている．

全体の結果として，既知ドメインで最も良い精度を達成したのは，モデルのパラメータサイズ T5-Base に対する shuf-shuf の生成結果であり，未知ドメインで最も良い性能を示したのは，モデルのパラメータサイズ T5-Large に対する shuf-shuf の生成結果である．これらのことから，レコード順序をシャッフルして Fine-tuning することが生成文の精度向上に効果的であることが分かる．

3.4　考察

前節で示した結果について，既知ドメインの結果では，シャッフルした順序で Fine-tuning することにより，順序に依存せず，各ドメインごとの汎化した文生成を行う能力を得ることが出来たといえる．

未知ドメインの場合では，モデルのパラメータサイズ T5-Small と T5-Base で各指標の向上が見られなかったが，これは学習したドメイン外の知識について，このパラメータサイズでは，事前学習および Fine-tuning 時に獲得することが出来ていなかったといえる．そのため，もともとの人為的な順序で Fine-tuning した def-def が，生成時にもその人為的な順序を手がかりとして生成した結果，shuf-shuf よりも高い精度を示したと考えられる．

一方で，モデルのパラメータサイズ T5-Large では，METEOR を除いて shuf-shuf が高い精度を達成している．これは T5-small および T5-Base とは反対に，パラメータサイズの大きい T5-Large では，事前学習および Fine-tuning の時点で未知ドメインへの表現能力を獲得したため，既知ドメインと同様に，シャッフルすることで順序に依存しない汎化した文生成が出来たと考えられる．

This work is licensed by the author(s) under CC BY 4.0
(https://creativecommons.org/licenses/by/4.0/).

表 2　レコード数 (n) ごとの BLEU 値

モデル		既知ドメイン							未知ドメイン				
Trainset	Testset	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$	$n=6$	$n=7$	$n=1$	$n=2$	$n=3$	$n=4$	$l=5$
T5-Small													
Default	Default	81.30	67.50	61.02	61.50	55.40	**64.10**	59.80	54.09	42.27	**37.99**	**38.61**	34.71
Default	Shuffle	81.30	67.37	59.20	61.35	54.36	63.05	**61.94**	54.09	**43.03**	37.81	38.40	**35.73**
Shuffle	Shuffle	**81.58**	**70.94**	**63.11**	**62.33**	**55.47**	63.73	61.29	**55.04**	41.56	37.06	37.43	34.49
T5-Base													
Default	Default	83.11	69.84	60.63	61.19	**56.30**	63.56	**62.92**	56.39	**48.77**	**45.78**	**43.79**	**40.50**
Default	Shuffle	83.11	68.78	60.13	61.17	54.75	61.48	60.02	56.39	46.82	44.05	43.15	38.74
Shuffle	Shuffle	**83.56**	**71.87**	**62.27**	**63.18**	56.06	**64.45**	61.98	**57.39**	46.03	43.06	43.10	38.78
T5-Large													
Default	Default	81.21	69.15	59.50	**61.20**	53.65	**67.46**	**60.06**	58.95	47.47	44.62	44.74	**41.04**
Default	Shuffle	81.21	67.12	59.01	60.07	54.61	66.83	59.46	58.95	46.65	44.06	43.99	40.80
Shuffle	Shuffle	**82.60**	**71.63**	**60.24**	60.44	**55.72**	65.43	59.67	**61.79**	**50.03**	**47.19**	**45.29**	39.90

表 3　レコード数 (n) ごとの件数と平均レコード数

n	Train	Test	Test $_{\text{既知}}$	Test $_{\text{未知}}$
1	4,345	1,089	545	544
2	3,368	976	473	503
3	3,638	1,055	523	532
4	3,394	987	482	505
5	2,398	708	359	349
6	488	58	58	-
7	471	55	55	-
Total	18,102	4,928	2,495	2,433
Ave.(n)	2.97	2.93	3.02	2.75

3.4.1　レコード数ごとの比較

各生成文を入力レコード数ごとに分け，個別に BLEU を計算した．表 2 に，その計算した BLEU 値を示す．また，学習用データおよびテストデータのレコード数ごとの件数と，その平均レコード数を表 3 に示す．

既知ドメインのレコード数 $n \leq 5$ のデータに対して，すべての設定において，shuf-shuf は def-def あるいは def-shuf よりも高い精度を示す傾向がみられた．一方で，既知ドメインのレコード数 $n = 6, 7$ では，def-def および def-shuf に対する shuf-shuf の有効性は陽に見られなかった．このことから，シャッフルして学習するだけでは，学習データの平均値の倍以上のレコード数をもつ入力に対しての表現方法を獲得することが出来ず，未知ドメインにおける T5-Small や T5-Base と同様に，もとの順序情報が生成の手助けをしたことで def-def の生成が高い精度を示したと考えられる．

未知ドメインのレコード数 $2 \leq n \leq 4$ では，表 1 の未知ドメインの結果に対する考察と同様の事が言える．しかし，レコード数 $n = 1$ では，T5-Small および T5-Base においても shuf-shuf が高い精度を示している．この結果から，もとのレコード順序で Fine-tuning することは，単一レコードに対する文生成の精度を犠牲にした上で複数レコードを持つ入力を学習していると考えられる．反対に，レコード数 $n = 5$ では，T5-Large においても shuf-shuf による精度向上が見られない．加えてレコード数 $n = 4$ についても，$1 \leq n \leq 3$ の精度向上に比べて低い．これらは，学習データの平均レコード数 $n = 2.97$ を中心にシャッフルによる汎化性能を獲得したため，既知ドメインのレコード数 $n = 6, 7$ と同様に，レコード数をもつ入力に対してはレコードの順序情報が文生成に大きく影響していると考える．

実験全体を通した結果から，レコード順序をシャッフルして入力することがデータ拡張になっているとも考えられ，今後さらに検証が必要である．

4　おわりに

本稿では，T5 を用いた Data-to-text モデルに対し，入力レコードの順序が生成文に及ぼす影響について検証した．自動評価指標による評価実験を通じて，T5 モデルが Fine-tuning 時にレコード順序も同時に学習することを確認し，モデルのパラメータサイズが十分大きい場合，レコード順序をシャッフルして Fine-tuning すると生成文の精度が向上することを確認した．今後は，Fine-tuning 時にエポックごとでレコード順序をシャッフルした場合の T5 モデルの挙動の検証や，T5 モデルおよび知識グラフを入力形式とする場合の適切な順序のプランニング手法の考案などをしていきたい．

謝辞

本稿の研究は，株式会社ウェザーマップの助成を受けた．

This work is licensed by the author(s) under CC BY 4.0
(https://creativecommons.org/licenses/by/4.0/).

参考文献

[1] Ehud Reiter and Robert Dale. *Building Natural Language Generation Systems*. Cambridge University Press, 2000.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. of NIPS 2017*, pp. 5998–6008, 2017.

[3] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, Vol. 21, No. 140, pp. 1–67, 2020.

[4] Mihir Kale and Abhinav Rastogi. Text-to-text pre-training for data-to-text tasks. In *Proc. of INLG 2020*, pp. 97–102, 2020.

[5] Leonardo F. R. Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. Investigating pretrained language models for graph-to-text generation. arXiv:2007.08426, 2020.

[6] Ratish Puduppully, Li Dong, and Mirella Lapata. Data-to-text generation with content selection and planning. In *Proc. of AAAI 2019*, pp. 6908–6915, 2019.

[7] Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. The WebNLG challenge: Generating text from RDF data. In *Proc. of INLG 2017*, pp. 124–133, 2017.

[8] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proc. of ACL 2002*, pp. 311–318, 2002.

[9] Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *Proc. of ICML 2018*, Vol. 80, pp. 4603–4611, 2018.

[10] Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proc. of EMNLP 2018*, pp. 66–71, 2018.

[11] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proc. of ACL 2016*, pp. 1715–1725, 2016.

[12] Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proc. of ACL 2018*, pp. 66–75, 2018.

This work is licensed by the author(s) under CC BY 4.0
(https://creativecommons.org/licenses/by/4.0/).