名詞句の並び替えによる教師なし言い換え生成の検討

杉浦 昇太　　西田 典起　　中山 英樹
東京大学 大学院情報理工学系研究科
{sugiura, nishida, nakayama}@nlab.ci.i.u-tokyo.ac.jp

1 はじめに

言い換え生成では，入力文と意味的に等価な文を自動的に生成することを目指す．言い換え生成は，情報検索やテキスト平易化, 機械翻訳のデータ拡張，文の前処理 (正規化) など, 自然言語処理の様々なタスクにおいて有益な技術になると期待されている [1].

従来技術は，質の高い言い換え生成器を訓練するために同じ意味となる文のペアを大量に必要とする [2]. そのような文のペアは自動的に収集することは難しく，人手によって大量にアノテーションするコストも大きい．また, データセットの多くは特定ドメインに偏りがちであり，たとえばベンチマークとして一般的に用いられている Quora Question Pairs [1]は疑問文の言い換え対のみを含んでおり，データ数もわずか 140k と大規模なデータセットとは言い難い．これらの問題は, 言い換え生成技術を多様なタスクに応用する際の大きな障壁となりうる．

そのような問題に対するアプローチとして，教師なし言い換え生成が研究されている．教師なし言い換え生成では，言い換えに関する教師データに依らず，生のテキストコーパスのみから言い換え生成器を構築することを目指す．Miao ら [3] らは，言い換え生成を単語単位の編集操作 (i.e., 挿入，削除，置換) の連続として定式化し，それらの操作を用いて文をサンプリングすることで言い換え生成を行う教師なし手法を提案した．しかし，それらの編集操作は局所的であるため, その生成結果もまた局所的な言い換えに留まるという問題がある．

そこで本研究では，既存手法では難しい態の交替のような文の構造レベルの言い換え生成を，教師なし手法によって行うことを目指す．本研究では，言い換え生成は次の三つのステップに分割できると考える．

(1) 入力文からの名詞句の抽出.

(2) 名詞句の並び替えとその間の表現の言い換え.

[1]https://www.kaggle.com/c/quora-question-pairs

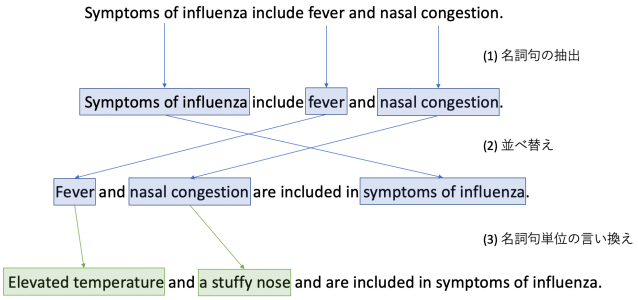

図 1: 提案手法の概要

(3) 名詞句単位の言い換え.

ステップ (1) は既存のチャンキングタスクであり，ステップ (3) はシソーラス等を活用することによって自動化できることが期待できる．一方，ステップ (2) の名詞句を並び替えたときの文の「糊付け」は，どのように行えるかが自明ではない．そこで本稿ではステップ (2) に焦点を当て，名詞句の順序に関する制約のもと，入力文と意味的に近い文を生成するモデルを提案する．具体的には，名詞句の順序がコントロールされた文の生成をリスト付き言語モデルによって実現し，それによって得られた言い換え候補の中で入力文と意味が類似しているものを言い換えとして出力する．提案手法を用いた実験を行い，得られた言い換え文を定性的に分析し，現状の課題について整理した．

2 関連研究

乾ら [1] は，スコープや必要な知識の種類によって言い換えを分類・整理し，それぞれの言い換えを計算機上で行う上での課題を指摘した．彼らは，形態・構文的な言い換えを自動で行うためには，言い換え規則の語彙依存性をうまく扱う必要があると述べた．本研究は，言い換えの語彙依存性を考慮した上で，名詞句の並び替えを通して形態・構文的な言い換えを目指すものであると解釈できる．

教師なし言い換え生成手法として，Variational Autoencoder (VAE) を用いて入力文に近い文を生成する手法 [4, 5] や，入力文に対する単語レベルの逐次的な編集操作を行うことで言い換えを生成する手法 [3, 6] が提案されている．しかし，いずれの手法も構造レベルで異なる言い換えを生成することは難しく，生成された文の多くは単語や句単位の局所的な言い換えに留まっている．本研究は，単語や句単位の局所的な言い換えではなく，文の構造のレベルから言い換えを行うことに焦点を当てている．

3 提案手法

本稿では，出力文における名詞句の順序をコントロールすることによって，既存手法では難しい態の交替のような構造レベルの言い換えの実現を目指す．

本手法における具体的な言い換え生成の手続きを以下に示す．

1. 入力文から名詞句を全て抽出する．
2. 名詞句の並び順を入れ替える．
3. 指定した並び順通りに名詞句を含む言い換え候補を複数生成する．
4. 生成された言い換え候補を意味類似度でリランキングし，上位のものを言い換え文として採用する．

名詞句の抽出は既存のチャンキング技術等を使うことによって容易に実現できる．そこで，以下ではステップ 3，ステップ 4 に対する提案手法について述べる．

3.1 名詞句の順序制約付き言語モデル

名詞句の順序制約付き言語モデルを作成し，名詞句を指定した順序で含むような文の生成を行う．提案手法では，まず入力文 x から名詞句を抽出し，その順序をリストとして表現する:

$$\mathrm{List}_t = [\mathrm{NP}_1, \mathrm{NP}_2, \ldots, \mathrm{NP}_m]. \tag{1}$$

ここで，List_t はステップ t におけるリストの状態を表し，NP_i はリストのトップから i 番目の名詞句を表す．各名詞句は単語列に分割されているとする．

名詞句のリストによって拡張した言語モデルによって文を生成する．提案手法による言語モデルでは，各

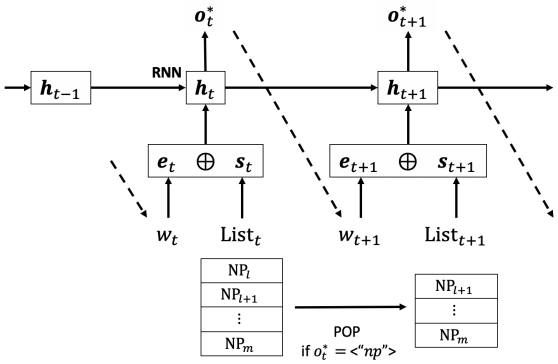

図 2: 名詞句の順序制約付き言語モデル

ステップにおいて，現状の出力文 (単語列) y_{t-1} とリスト List_t の中身の情報から，リスト先頭の名詞句 $\mathrm{NP}_{\mathrm{top}}$ を出力するか，辞書中の単語 $w_t \in V$ を出力するかを決定する:

$$o_t^* = \underset{o_t \in V \cup \{\text{“}\langle \mathrm{np} \rangle\text{”}\}}{\operatorname{argmax}} P(o_t \mid y_{t-1}, \mathrm{List}_t). \tag{2}$$

ここで o_t^* はステップ t における出力記号であり，$o_t^* = w_t \in V$ のとき，単語 w_t を出力文の末尾に結合する．これは通常の言語モデルの処理に等しい．“⟨np⟩” は本研究で導入する特別記号であり，$o_t^* =$ “⟨np⟩” のとき，リストの先頭から名詞句 $\mathrm{NP}_{\mathrm{top}}$ をポップし，$\mathrm{NP}_{\mathrm{top}}$ を出力文の末尾に結合する:

$$y_t = \begin{cases} y_{t-1} \cdot w_t & (o_t^* = w_t \in V \text{ のとき}) \\ y_{t-1} \cdot \mathrm{NP}_{\mathrm{top}} & (o_t^* = \text{“}\langle \mathrm{np} \rangle\text{” のとき}) \end{cases}. \tag{3}$$

リストが空のとき，$P(o_t = \text{“}\langle \mathrm{np} \rangle\text{”} \mid y_{t-1}, \mathrm{List}_t) = 0$ とする．リストが空であり，かつ $o_t^* =$ “⟨EOS⟩” のとき，出力文の生成を終了する．

本稿では recurrent neural network [7] を用いて上記の言語モデルを構築する．提案モデルのネットワーク構造を図 2 に示す．RNN 言語モデルの隠れ状態を $\boldsymbol{h}_t$ とすると，それを次のように更新する:

$$\boldsymbol{h}_t = \mathrm{RNN}(\boldsymbol{h}_{t-1}, \boldsymbol{e}_t \oplus \boldsymbol{s}_t). \tag{4}$$

ここで $\boldsymbol{e}_t$ は直前に出力した単語の埋め込み，$\boldsymbol{s}_t$ はリスト List_t の中身の特徴ベクトルであり，リストに含まれる名詞句を結合して得られる単語列 $\tilde{w}_1, \ldots, \tilde{w}_l$ に対して前方向 RNN，後方向 RNN それぞれを適用し

たときの最後の隠れ状態の和とする:

$$\boldsymbol{s}_t = \boldsymbol{f}_l + \boldsymbol{b}_1, \tag{5}$$

$$\boldsymbol{f}_1, \ldots, \boldsymbol{f}_l = \overrightarrow{\mathrm{RNN}}(\tilde{w}_1, \ldots, \tilde{w}_l), \tag{6}$$

$$\boldsymbol{b}_1, \ldots, \boldsymbol{b}_l = \overleftarrow{\mathrm{RNN}}(\tilde{w}_1, \ldots, \tilde{w}_l). \tag{7}$$

名詞句を結合して単語列にする際に，名詞句の切れ目を示す記号 “⟨sep⟩” を名詞句間に挿入する．演算子 $\oplus$ はベクトルの結合を表す．

本稿で提案する名詞句の順序に関する制約付き言語モデルは，生のテキストコーパスのみから学習することが可能であり，教師なし手法である．各ステップにおいて正しい出力記号を生成するように，交差エントロピーを損失関数とする学習を行う．

3.2 意味的類似度に基づく候補文のリランキング

3.1 節で導入した制約付き言語モデルによって，名詞句を指定した順序で含む文を生成することができる．しかし，たとえ入力文中に現れる名詞句をすべて含んでいるとしても，出力文が入力文と意味的に等価な文になる保証はない．

そこで本研究では，上記の制約付き言語モデルで生成した言い換え候補を，入力文との意味的類似度に基づいてリランキングし，類似度の高いものを言い換え文として出力する．文間の意味的類似度は， 入力文と出力候補文それぞれの文ベクトルのコサイン類似度とする．文のベクトル表現を計算するために，本稿では教師なし文表現手法の一つである Sent2Vec [8] を用いた．

4 実験

データセット　学習コーパスとして，MS-COCO[9] に含まれる画像説明文を用いた．MS-COCO に含まれる画像説明文のうち 546,605 文を制約付き言語モデルの訓練に，24,818 文を検証に使用し，学習に使用しなかった 24,808 文に対して提案手法による言い換え生成を行った．

実験設定　名詞句の抽出は句構造解析器を適用することで行った．句構造解析には spaCy [2]を使用した．RNN 言語モデルとしては 2 層の LSTM を使用し，単語ベクトルおよび RNN の隠れ層の次元数はともに 256 次元とした．言い換え生成時のビーム幅は 10 とした．入力文に含まれる名詞句の並び替えを列挙し，その各々について言い換えを生成した．

実験結果　表 1 に，提案手法によって生成された言い換え文を載せる．名詞句の順序に関する制約に従い，各名詞句の間を適切な表現で埋めた流暢な文が生成されていることがわかる．

一方で，流暢ではあるが，入力文との意味を保存できず，適切な言い換えが生成できなかったケースも見られた．例えば，表 1 の (4a) と (5a) は，入力文と意味的に等価な文を生成できていない例である．

5 考察

入力文の意味の保存　提案手法では，言い換え候補を複数生成し，それらを意味類似度でリランキングすることで，入力文と意味が近い文を得ることを試みた．この方法により正しい言い換えを得るためには，言い換え候補の中に入力文と意味の近い文が含まれている必要がある．しかし，実験結果を観察してみると，そもそも候補に入力文と意味が等価な文が含まれていない場合が多くみられた．

そこで，より適切な言い換えを生成するためには，候補文の生成時に入力文との意味の類似度を適切に考慮するような手法を考える必要があり，これは今後の課題である．

名詞句の並び替えによって得られる言い換え文の性質　名詞句の並び替えを行う提案手法では，統語的および形態構文的な言い換えが実現できると期待される．実際，結果を観察してみると，一部の出力文 において副詞句の移動 (e.g., (1a)) や態の交替 (e.g., (2a)) が生じていることが確認できた．

一方で，本手法では実現不可能な形態構文的な言い換えが存在することも明らかになった．たとえば，代名詞の語形変化や名詞句の省略を伴うような言い換え (e.g., “**He** bought the car” → “The car was bought by **him**”) は生成できず，意味の異なる文が出力される．これを踏まえ，今後は幅広い構文的な言い換えを生成できる手法を模索したい．

[2]https://spacy.io/

表 1: 提案手法によって生成した言い換え文．下線は抽出された名詞句を表す．下線に添字がある場合は，リスト中での順番を表す．

入力	a man is in mid air doing a skateboard trick
出力 (1a)	a man$_1$ is doing a skateboard trick$_2$ in mid air$_3$
出力 (1b)	a skateboard trick$_1$ performed by a man$_2$ in mid air$_3$
入力	the people fly a kite by the tall stone building
出力 (2a)	a kite$_1$ is being flown by the people$_2$ by the tall stone building$_3$
出力 (2b)	the people$_1$ are standing by the tall stone building$_2$ flying a kite$_3$
入力	a bed filled with different types of stuffed animals
出力 (3a)	different types$_1$ of stuffed animals$_2$ on a bed$_3$
出力 (3b)	a bed$_1$ filled with stuffed animals$_2$ of different types$_3$
入力	a man tosses a frisbee across to his friend
出力 (4a)	his friend$_1$ watching as a man$_2$ catches a frisbee$_3$
出力 (4b)	a man$_1$ with his friend$_2$ throwing a frisbee$_3$
入力	a family kitchen cluttered with items on the counters
出力 (5a)	items$_1$ in a family kitchen$_2$ on the counters$_3$
出力 (5b)	the counters$_1$ of a family kitchen$_2$ are filled with items$_3$

言い換え生成の定量的評価　適切な定量的評価による提案手法の検証は今後の課題である．提案手法を適切に評価するためには，入力文と生成文が同じ意味であるかという従来の評価尺度に加えて，それらが構造的に異なるかどうかを評価する手法を考える必要がある．

6　おわりに

本稿では，文の構造レベルの言い換え生成を教師なし手法により実現することを目指し，名詞句順序の制約付き言語モデルとそれを用いた言い換え生成手法を提案した．実験により得られた言い換え文を定性評価し，提案手法の問題点を整理した．

謝辞

本研究は JSPS 科研費 JP19K22861 の助成を受けたものです．

参考文献

[1] 乾 健太郎，藤田 篤. 言い換え技術に関する研究動向. **自然言語処理**, 11(5):151–198, 2004.

[2] Aaditya Prakash, Sadid A. Hasan, Kathy Lee, Vivek Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. Neural paraphrase generation with stacked residual LSTM networks. In *COLING*, 2016.

[3] Ning Miao, Hao Zhou, Lili Mou, Rui Yan, and Lei Li. CGMH: Constrained sentence generation by metropolis-hastings sampling. In *AAAI*, 2019.

[4] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *CoNLL*, 2016.

[5] Aurko Roy and David Grangier. Unsupervised paraphrasing without translation. In *ACL*, 2019.

[6] Xianggen Liu, Lili Mou, Fandong Meng, Hao Zhou, Jie Zhou, and Sen Song. Unsupervised paraphrasing by simulated annealing. *ArXiv*, abs/1909.03588, 2019.

[7] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH*, 2010.

[8] Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. Unsupervised learning of sentence embeddings using compositional n-gram features. In *NAACL-HLT*, 2018.

[9] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014.