大規模言語モデルに対する語彙置換継続事前学習の有効性の検証

野崎雄太 [1] 中島大 [1] 佐藤諒 [1] 伊藤真也 [1] 近藤宏 [1] 麻場直喜 [1] 川村晋太郎 [1]
[1] 株式会社リコー デジタル戦略部 デジタル技術開発センター 言語 AI 開発室
{yuta.nozaki1,dai.nakashima,ryo.sato4,shinya.itoh,hiroshi.xx.kondoh,
naoki.asaba,shintaro.kawamura}@jp.ricoh.com

概要

英語モデルに日本語データを継続事前学習することによって日本語性能を高めるというアプローチにおいて，最も重要な課題の１つに日本語に最適化されていないトークナイザーによる学習効率の低下が挙げられる．トークナイザーの語彙拡張など様々な対策手法があるが，モデルパラメータの増大という副作用もある．そこで本研究ではトークナイザーの語彙拡張をせずに，既存モデルの埋め込みを活用することによって学習効率と精度を高める手法，語彙置換継続事前学習を提案する．検証の結果，語彙拡張と同等の精度が確認された．

1 はじめに

近年 Transformer [1] ベースの大規模言語モデルの研究が加速し，様々な LLM が公開されている．しかし，その大部分 [2, 3] は大量の英語テキストによって学習された英語特化モデル (英語モデル) であり，日本語に特化したモデル (日本語モデル) は少ないのが現状である．主な理由として，日本語の良質な大規模コーパスは英語と比較して少量であり，学習データ量を増やすことで言語モデルとしての性能が向上するという法則 [4] の上で制約となっていることが考えられる．

いわゆる低リソース言語に特化したモデルを構築する場合，十分なデータ量で学習された英語モデルにその言語データを継続事前学習させるというアプローチがある [5, 6]．日本語の事例でも英語モデルに日本語データを継続事前学習することで日本語モデルを構築したという報告がある [7]．しかしこの場合，既存のモデルはトークナイザーの語彙の大半が英語であるため，このまま日本語データを継続事前学習すると同一バッチ内で学習できる文字列長が英語と比較して少なくなってしまい，学習効率が非常に悪い．そのため日本語の語彙が多く含まれるトークナイザーを構築して学習を行うことが求められる．

トークナイザーに変更を加えて継続事前学習を行う手法には以下のようなものが挙げられる．

- Scratch: 新規トークナイザーで継続事前学習
- Extend: トークナイザー語彙を拡張して継続事前学習

Scratch はトークナイザーの語彙と既存モデルの語彙の埋め込みベクトルの一致率が著しく低くなり，既存の重みを活用できず，精度が低くなってしまう可能性がある．一方で Extend は拡張された語彙の埋め込みベクトルの初期値を工夫することによって既存モデルの重みを活用しながら日本語に特化したトークナイザーで継続事前学習ができる．BERT [8] などのエンコーダモデルに対する低リソース言語の継続事前学習 [6, 9] やドメイン適応の分野でもこの手法は一般的である [10, 11, 12, 13]．しかし，トークナイザーの語彙を拡張すると，同時にモデルサイズが語彙数に応じて増加し，最終的に GPU メモリや学習時間などコストが増大する．また，語彙数を一定以上増やしても性能は飽和するという報告もある [14]．これらの問題を解決するために，本研究では既存トークナイザーの語彙の一部を置換することで，既存の埋め込みを活用しながら語彙数を拡張せずに継続事前学習を行う手法，語彙置換継続事前学習を提案する．具体的には，新しくトークナイザーを構築し，既存トークナイザーのみに含まれる語彙を新規トークナイザーのみに含まれる語彙に置換する．また，置換した語彙に対応するモデルの埋めこみベクトルはその語彙を構成するトークン列の埋め込みベクトルの平均に置換する．これによっ

て語彙数を増やさずに既存モデルの埋め込みを活用しながら継続事前学習を行うことができる．

2 関連研究

[15] はトークナイザーの効率を評価する指標として，トークンあたりの文字列長 (length per token) を挙げており，多言語トークナイザーを構築する場合，一定の語彙数の水準で length per token が飽和することを確認した．本研究ではこの事例をベースに英語語彙数が減少したことによる英語タスク精度げの影響の検証も行う．

3 語彙置換継続事前学習

語彙置換継続事前学習は語彙の置換と埋め込みの置換の 2 つの手法を組み合わせている．図 1 に概略を示す．

3.1 語彙の置換

トークナイザー T を語彙 V と語彙から語彙 ID をマッピングする関数 f の組 (T, f) と定義し，既存のトークナイザー T_A と新規トークナイザー T_B とする．それぞれのトークナイザーは語彙 V_A，V_B で同じ語彙数を持つ．

$$\begin{aligned} V_A &= \{v_1^A, v_2^A, \ldots, v_n^A\} \\ V_B &= \{v_1^B, v_2^B, \ldots, v_n^B\} \end{aligned} \tag{1}$$

T_A の各語彙 v_i^A に対する語彙 ID k_i^A をマッピングする関数 f_A，同様に T_B の各語彙 v_1^B に対する語彙 ID k_i^B をマッピングする関数 f_B を定義する．

$$f_A(v_1^A) = k_i^A, \quad f_B(v_i^B) = k_i^B \tag{2}$$

V_A と V_B の共通語彙を V_{com}，V_A の独自語彙を V_{uniA}，V_B の独自語彙を V_{uniB} とする．

$$\begin{aligned} V_{\text{com}} &= V_A \cap V_B, \\ V_{\text{uniA}} &= V_A - V_B, \quad V_{\text{uniB}} = V_B - V_A \end{aligned} \tag{3}$$

V_{uniA} の全ての語彙 v_i^A を V_{uniB} の語彙 v_i^B に置換し，置換済み語彙集合を V_A' とする．また，新しいマッピング関数 $f_{A'}$ を定義する．$f_{A'}$ は T_B の各語彙 v_i^B に対して，T_A の対応する語彙 ID k_i^A を割り当てる．

$$f_{A'}(v_i^B) = k_i^A \tag{4}$$

V_A' と $f_{A'}$ を持つトークナイザーを語彙置換トークナイザー $T_{A'}$ とする．$T_{A'}$ は T_A をベースとしていることから，$f_A(V_{\text{com}}) = f_{A'}(V_{\text{com}})$ となる．つまり T_A で学習したモデルの埋め込みベクトルとの対応関係も維持される．

3.2 埋め込みの置換

3.1 で置換したトークナイザーの語彙は既存モデルの埋め込みベクトルとは意味的な対応関係がない．そのため先行事例 [11] と同様に，既存モデルのトークン列の平均ベクトルで埋め込みベクトルも置換する．具体的には，語彙を辞書としてテキストをトークン分割しトークン列 L を返す関数 $Tokenize(V, text)$ を定義し，既存トークナイザー T_A の語彙 V_A を辞書として，$v_i^B \in V_{\text{uniB}}$ をトークン分割し，トークン分割されたトークンを $S_{v_i^B}$ とする．$S_{v_i^B}$ の各トークンに対する既存モデルの埋め込みベクトル $\overrightarrow{S_{v_i^B}}$ を取得し，平均埋め込みベクトルを計算する．この平均埋め込みベクトルは v_i^B の意味を表すベクトル $\overrightarrow{v_i^B}$ となる．

$$\overrightarrow{v_i^B} = \frac{1}{|S_{v_i^B}|} \sum \overrightarrow{S_{v_i^B}} \tag{5}$$

既存モデルの埋め込みベクトル集合 $\overrightarrow{V_A}$ において，$v_i^B \in V_{\text{uniB}}$ の各トークンに対応する埋め込みベクトルは語彙 v_i^A の意味を持つベクトル $\overrightarrow{v_i^A}$ である．そのため $\overrightarrow{V_A}$ のうち，$v_i^A \in V_{\text{uniA}}$ となる全ての $\overrightarrow{v_i^A}$ を，$\overrightarrow{v_i^B}$ に置換し，置換済み埋め込みベクトル集合 $\overrightarrow{V_{A'}}$ とする．この $\overrightarrow{V_{A'}}$ は語彙置換トークナイザー $T_{A'}$ の語彙 $V_{A'}$ の意味がより正確に表現されているベクトルであると考えられ，学習がより効率的になると考えられる．

4 検証実験

語彙置換のみを適用して継続事前学習した手法 (Replace) と埋め込み置換も行った手法 (Embedding-Replace) の有効性を検証する．

4.1 実験設定

実験で使用するベースとなる既存トークナイザーおよびモデルは Llama 2 7B Chat [3] とした．新規トークナイザーは日本語データ（Wikipedia・CC-100）と英語データ（Wikipedia）を 7 : 10 の割合でサンプリングしたデータから構築し，提案手法を用いて語彙置換トークナイザー $T_{A'}$ を構築した．比較手法として，既存トークナイザー T_A を使った手法 (Keep)，新規トークナイザー T_B を使って継続事前学習した手法 (Scratch)，語彙拡張トークナイザー T_E を使った手法 (Extend) を設定する．Extend は先行研究 [11] の手法に準拠した．$V_A \cup V_B$ で得られる語彙集合を V_E とし

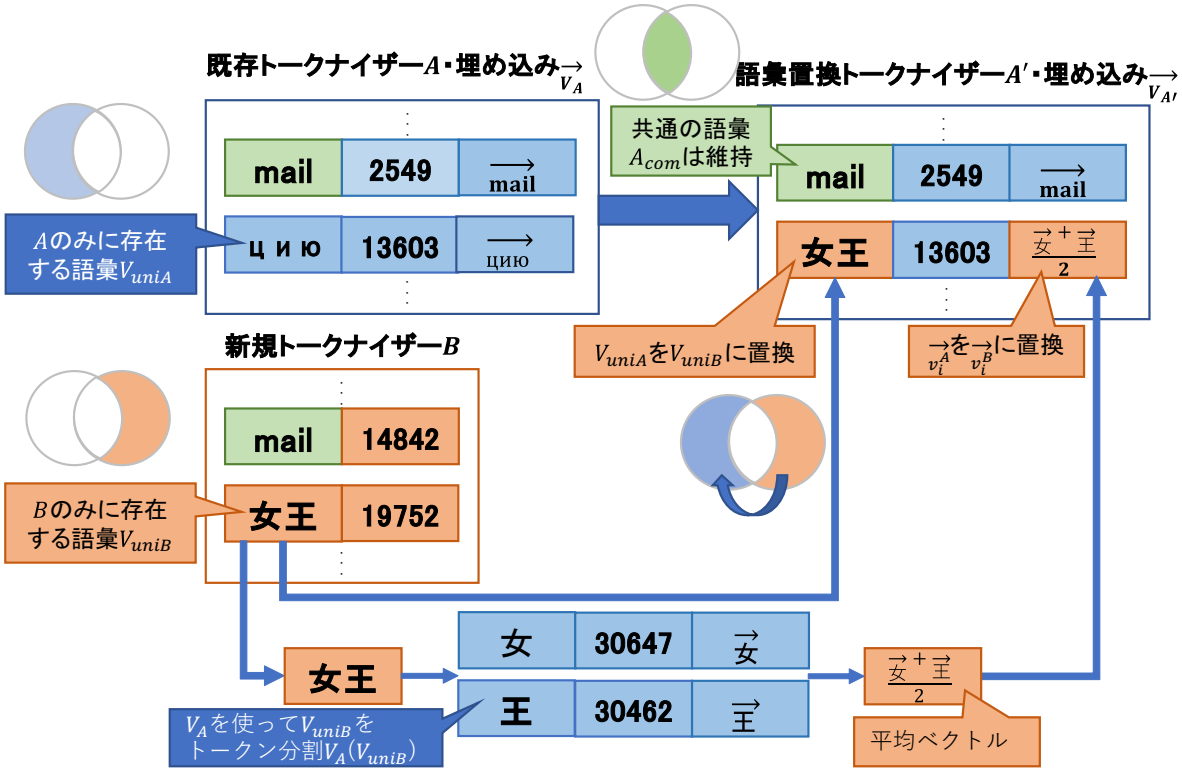

図 1　語彙置換継続事前学習の概略図

て，$|V_E|$ まで埋め込みベクトルの数を増加させた．トークナイザー A によって V_E をトークナイズし，その埋め込みベクトルの平均を埋め込みベクトル $\overrightarrow{V_E}$ として設定した．

これらの設定でモデルを学習した．また，継続事前学習データは日本語データ（Wikipedia）を利用した．

4.2　トークナイザー評価

既存・新規・語彙置換・語彙拡張トークナイザーのそれぞれの日本語及び英語語彙数と 1 トークンあたりの平均文字数 (length per token) を検証した．日本語の検証データは CC-100 データセット，英語は mC4 データセットを使用した．結果を表 1 に示す．既存と比較すると語彙置換や語彙拡張は日本語では約 2 倍となっており，語彙拡張が最も length per token の値が高かったが，語彙置換と比較して大きな差はなかった．また，英語では語彙置換は英語語彙数が少なくなるため，length per token の値は低くなることが予想されるが，大きく低下していないことが確認された．これは [15] と同様の結果と同じことを示している．これによって語彙拡張でなくても，同水準に推論効率を向上できることが確認された．また，語彙と埋め込みベクトルの一致率を確認した．一致率が高いほど既存のモデルの埋め込みを直接活用できることを表す．結果を表 1 に示す．一致率は 35.6 %，拡張は 60.82 % であった．

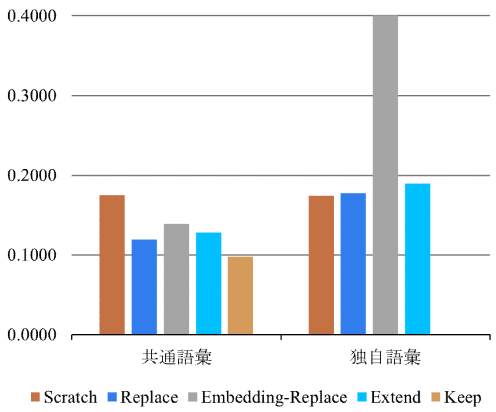

図 2　埋め込みベクトルの変化率の平均値

4.3　モデル評価

続いて，日本語および英語タスクによる精度評価，共通語彙の埋めこみ及び独自語彙の埋め込みの変化率 (学習前後のベクトルの L2 ノルムの平均) を比較する．

最初にモデル学習後の日本語タスクの精度比較を行う．表 2 に各タスクにおける精度を示す．結果は Keep が一部のタスクを除き，最

表 1 トークナイザーの統計

	語彙数	日本語語彙	英語語彙	埋め込み一致語彙数	埋め込み一致語彙率 (%)	length per token JP	length per token EN
既存 T_A	32000	834	27667	32000	100	0.85	4.01
新規 T_B	32000	17030	13839	266	0.83	2.01	3.99
語彙置換 $T_{A'}$	32000	17030	13839	11382	35.6	2.01	3.99
語彙拡張 T_E	52618	31311	17165	32000	60.82	2.08	4.20

表 2 日本語タスク，英語タスクの精度評価結果：タスク名 (ショット数, 評価指標)

言語	日本語						英語			
タスク	JCommonsenseQA (3,acc_norm)	JSQuAD (3,F1)	JAQKET (3,F1)	JaQuAD (3,F1)	JEMHopQA (3,F1)	XL-Sum-ja (3,R-2)	ARC (25, acc_norm)	HellaSWAG (10,acc_norm)	MMLU (5,acc)	WinoGrande (5,EM)
Llama-2	36.55	81.76	49.37	70.49	40.09	6.07	72.55	76.68	45.25	66.46
Keep	34.85	84.34	58.89	77.74	43.34	6.77	69.81	75.41	45.65	67.17
Scratch	29.22	66.52	51.66	64.35	21.21	7.36	26.44	27.83	25.98	47.91
Replace	36.91	81.87	72.03	74.88	39.78	8.47	56.4	62.45	41.44	64.09
Embed-Rep	38.7	83.81	74.50	76.79	39.9	9.75	61.39	68.6	43.33	65.98
Extend	39.77	82.11	74.99	75.90	39.41	11.31	62.68	69.59	43.58	66.93

も精度が良いことが確認された．また，同時に Embedding-Replace と Extend が同水準の精度であることが確認され，語彙置換継続事前学習の有効性が確認された．一方で Scratch は他の手法と比較してタスクの精度が大幅に落ちていることが確認された．これは埋め込みとトークナイザーの語彙が対応していないことが原因と考えられる．この結果により，語彙置換手法の有効性が確認された．また，Embedding-Replace と Replace を比較すると，Embedding-Replace の精度が良いことから，埋め込み置換手法の有効性も確認された．

次に英語タスクの精度比較を行う．本実験では日本語データのみで継続事前学習を行ったため，英語を忘却し，英語タスクの精度低下が想定される．また，語彙置換トークナイザーは英語語彙数を減少させているので，Embedding-Replace や Replace で破滅的な忘却が発生する可能性がある．英語の各タスクを用いて Extend と比較した精度の低下を検証した．検証の結果，Embedding-Replace は Extend と同等の水準の精度であることが確認された．このため，語彙数を減少したことによる大幅な精度の低下は見られないことが確認された．

続いて埋め込みベクトルの変化率の評価を行う．図 2 に結果を示す．Keep の埋め込み変化率に比べて，その他の手法は共通語彙，独自語彙共に変化率が高いことが確認された．Scratch では 266 件の共通語彙のほとんどが意味を持たないトークンであったため，独自語彙と同程度の変化率となったと推察される．Replace や Embedding-Replace，Extend では概ね独自語彙の変化率が高く，共通語彙の変化率が低いことが確認された．一方で Embedding-Replace の独自語彙の変化率が他の手法と比較して非常に高いことが確認されている．

5 おわりに

本論文では語彙数を増加させず，語彙や埋め込みの置換を行って継続事前学習を行う手法を提案した．検証実験の結果，語彙置換継続事前学習の有効性が確認され，語彙拡張手法と同水準の性能であることが確認された．また，共通語彙および独自語彙間で埋め込みベクトルの変化率に一定の差があることが確認された．

今後の展望として，語彙と埋め込みベクトルの一致率と精度の相関関係について調査を行うことが挙げられる．

本研究の成果を用いて，より大規模に継続学習を行い，日英モデルを構築した [16].

参考文献

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. **Advances in neural information processing systems**, Vol. 30, , 2017.

[2] OpenAI, :, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, et al. Gpt-4 technical report, 2023.

[3] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models, 2023.

[4] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020.

[5] Zheng Xin Yong, Hailey Schoelkopf, Niklas Muennighoff, Alham Fikri Aji, David Ifeoluwa Adelani, Khalid Almubarak, M Saiful Bari, Lintang Sutawika, Jungo Kasai, Ahmed Baruwa, , et al. BLOOM+1: Adding language support to BLOOM for zero-shot prompting. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, **Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**, pp. 11682–11703, Toronto, Canada, July 2023. Association for Computational Linguistics.

[6] Zihan Wang, Karthikeyan K, Stephen Mayhew, and Dan Roth. Extending multilingual BERT to low-resource languages. In Trevor Cohn, Yulan He, and Yang Liu, editors, **Findings of the Association for Computational Linguistics: EMNLP 2020**, pp. 2649–2656, Online, November 2020. Association for Computational Linguistics.

[7] elyza. elyza/elyza-japanese-llama-2-7b · hugging face, 2023. https://huggingface.co/elyza/ELYZA-japanese-Llama-2-7b.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, **Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)**, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[9] Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. UNKs everywhere: Adapting multilingual language models to new scripts. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, **Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing**, pp. 10186–10203, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[10] Arata Suganami and Hiroyuki Shinnou. Construction of Japanese BERT with fixed token embeddings. In Shirley Dita, Arlene Trillanes, and Rochelle Irene Lucas, editors, **Proceedings of the 36th Pacific Asia Conference on Language, Information and Computation**, pp. 356–361, Manila, Philippines, October 2022. Association for Computational Linguistics.

[11] Yunzhi Yao, Shaohan Huang, Wenhui Wang, Li Dong, and Furu Wei. Adapt-and-Distill: Developing Small, Fast and Effective Pretrained Language Models for Domains. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, **Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021**, pp. 460–470, Online, August 2021. Association for Computational Linguistics.

[12] Hirotaka Tanaka and Hiroyuki Shinnou. Vocabulary expansion of compound words for domain adaptation of bert. In **Proceedings of the 36th Pacific Asia Conference on Language, Information and Computation**, pp. 379–387, 2022.

[13] 照乃梶浦, 美唯佐藤, 菜生相馬, 舞衣高橋, 百々雅小原, 君郎倉光. 多言語事前学習済みモデル mt5 への python 言語モデルの追加学習. 情報処理学会論文誌プログラミング（PRO）, Vol. 16, No. 2, pp. 29–29, 06 2023.

[14] Wenlin Chen, David Grangier, and Michael Auli. Strategies for Training Large Vocabulary Neural Language Models. In Katrin Erk and Noah A. Smith, editors, **Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**, pp. 1975–1985, Berlin, Germany, August 2016. Association for Computational Linguistics.

[15] 中島大, 野崎雄太, 佐藤諒, 麻場直喜, 川村晋太郎, (in press). BPE を用いたトークナイザーの性能に対する, 言語・語彙数・データセットの影響. 言語処理学会 第 30 回年次大会 発表論文, 2024.

[16] 麻場直喜, 野崎雄太, 中島大, 佐藤諒, 池田純一, 伊藤真也, 近藤宏, 小川武士, 坂井昭一朗, 川村晋太郎 press). 語彙置換継続事前学習による日英バイリンガルモデルの構築と評価. 言語処理学会 第 30 回年次大会 発表論文, 2024.