同一条件下における Encoder/Decoder アーキテクチャによる文埋め込みの性能分析

岡田龍樹　杉本徹
芝浦工業大学
{al22032, sugimoto}@shibaura-it.ac.jp

概要

Retrieval-Augmented Generation (RAG) の実用化が進む中，文埋め込みモデルの重要性が増している．近年，大規模事前学習済み Decoder モデルを文埋め込みに転用する手法が提案されているが，従来の Encoder モデルとの性能・実用性の比較は十分でない．本研究では，同一のパラメータ数・学習データ・学習手法で構築した Encoder モデルと Decoder モデルを定量的に比較した．その結果，同一条件下では Encoder モデルが下流タスク性能の面で Decoder モデルを上回った．また処理速度やメモリ効率の面でも優位であることを示した．一方，大規模事前学習済み Decoder モデルの知識転用は，追加コストを抑えつつ性能向上をもたらす有効な手法であることも確認された．これらの知見は，実用的な文埋め込みモデルの選択指針を提供するだけでなく，アーキテクチャ選択が性能に与える影響の理解を深める重要な示唆となる．

1　はじめに

自然言語生成タスクにおいて，外部コーパスを検索し回答を生成する Retrieval-Augmented Generation (RAG) [1] が注目を集めている．RAG では検索時において，文やクエリをベクトル化する文埋め込みモデルの性能がシステム全体に影響するため，優れた文埋め込みモデルが不可欠である．

BERT [2] など従来用いられてきた Transformer の Encoder モデルに対し，LLM2Vec [3] を代表とする Llama [4] など Decoder モデルを用いた学習手法が提案されている [5, 6]．これは大規模事前学習済みモデルを用い，知識転用による性能向上が見込めることへの期待である．しかし，この学習済みモデルはパラメータ数や学習データが従来の Encoder モデルと大きく異なり，利点や実用性が明らかでない．

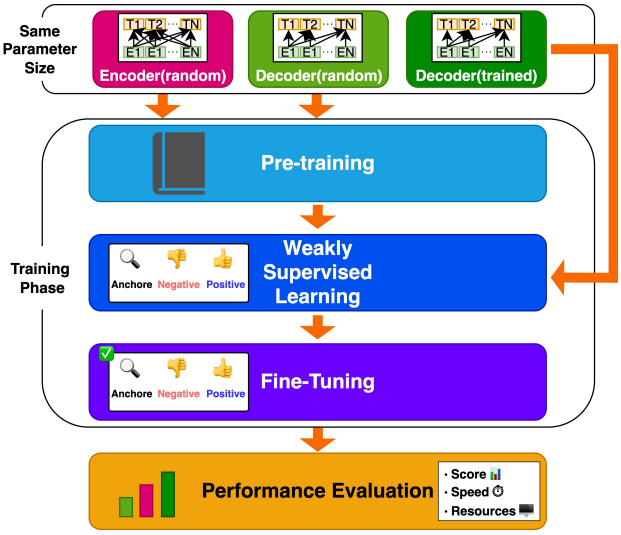

図 1　研究の全体像．同一条件（パラメータ数・学習データ）下での Encoder・Decoder モデルと，学習済み Decoder モデルをベースとした文埋め込みモデルの比較．

本研究では，Encoder と Decoder アーキテクチャの違いが文埋め込み性能に与える影響を明らかにし，実用的な選択指針を提供するため，同一条件下で両モデルを構築・学習し，性能・計算効率・文埋め込み表現の品質の観点から定量的に比較を行う．図 1 に研究の全体像を示す．同一のパラメータ数・学習データで構築した Encoder モデルと Decoder モデル，さらに大規模事前学習済み Decoder モデルを比較することで，アーキテクチャの違いと知識転用の効果を明らかにする．

具体的には，以下の 3 つの観点から評価を行う．

- 下流タスクにおける文埋め込み性能の比較
- 運用時の処理速度とメモリ効率の比較
- 埋め込み表現の品質評価

また，本研究で用いたソースコード[1)]とモデルチェックポイント[2)]を公開している．

1）https://github.com/iamtatsuki05/MIREI
2）https://huggingface.co/collections/iamtatsuki05/mirei

2 関連研究

2.1 文埋め込みモデルの構築手法

文埋め込みモデルの構築には，対照学習が広く用いられている．E5 を代表に，タイトルと本文などのペアデータで弱教師あり学習 (WSL) の後，クエリ・正例・負例を用いた教師あり学習 (FT) を行う 2 段階での対照学習が一般的である [7, 8, 9]．日本語においては，Ruri [10] がこれらの手法を参考とした日本語データセット・モデルを提案している．

2.2 Decoder モデルによる文埋め込み

LLM2Vec [3] を代表に，Decoder モデルを文埋め込みに転用する手法が提案されている [5, 6]．LLM2Vec では，Decoder モデルの Attention を双方向化し，対照学習によって埋め込み能力を獲得させる．しかし，これらの研究は Encoder モデルとのパラメータ規模の乖離，E5 等の一般的な手法と異なる学習設定，大規模事前学習済みモデルを用いることによる知識転用の効果が不明という課題がある．

2.3 文埋め込みモデルの評価方法

文埋め込みモデルの評価には，検索や分類など複数のタスクで構成される MTEB [11] ベンチマークが主流である．日本語では，JMTEB [12] が 5 タスク・28 データセットで構成され，日本語の文埋め込みモデルを総合的に評価できる．また，埋め込み表現の品質を測定する指標として，Alignment と Uniformity [13] が提案されている．Alignment は類似サンプルが特徴量空間上で近くに分布しているかを，Uniformity は特徴表現が単位超球面上に一様に分布しているかを測定し，SimCSE [14] などの対照学習手法の評価にも広く用いられている．

3 実験設計

3.1 設計方針

本研究では，Encoder と Decoder アーキテクチャの文埋め込み性能への影響を純粋に評価するため，以下の条件を可能な限り統一した実験設計を行う．

- non-embedding parameters を 0.5B で統一
- 同一の sarashina2.2-0.5b トークナイザーを使用
- 同一の学習データセットを使用
- 同一の最適化設定を使用

これにより，モデルの規模，トークナイザー，学習データ，最適化手法といったアーキテクチャ以外の変数を極力排除し，Encoder・Decoder 間の性能差が定量的に検証可能となる条件を整える．さらに，大規模事前学習済み Decoder モデルを加えることで，知識転用の効果についても検証する．

3.2 モデル構成

本研究では，以下の 3 つのモデルを構築・評価する．

(1) Sentence-ModernBERT-JP-0.5B：ModernBERT [15] ベースの 0.5B パラメータの Encoder モデル．マスク率 30% のマスク穴埋めタスクで事前学習した後，文埋め込み学習を行う．

(2) Sentence-Llama-Bi-JP-0.5B：Llama ベースの 0.5B パラメータの Decoder モデル．次単語予測タスクで事前学習した後，文埋め込み学習を行う．文埋め込み学習時には，LLM2Vec に準じて Attention を双方向化する．

(3) Sentence-Sarashina-Bi-0.5B：約 10T トークンの日本語・英語コーパスで事前学習済みの sarashina2.2-0.5b[3] を用いた 0.5B パラメータの Decoder モデル．LLM2Vec に準じて Attention を双方向化し，文埋め込み学習を行う．

3.3 データセット

事前学習では，fineweb-2-edu-japanese[4] を最大系列長 512 トークンで 10B トークン分，wiki40b_ja[5] を最大系列長 8192 トークンで 1B トークン分 2 段階で学習した．文埋め込み学習では，WSL に ruri-dataset-v2-pt[6] を，FT に ruri-v3-dataset-ft[7] を用いて 2 段階対照学習を実施した．

3.4 学習設定

事前学習と文埋め込み学習の詳細な設定は付録 A に記載する．学習量や最大学習系列長，最適化条件などのハイパーパラメータは ModernBERT, Chinchilla [16]，E5 を参考に設定した．

3) https://huggingface.co/sbintuitions/sarashina2.2-0.5b
4) https://huggingface.co/datasets/hotchpotch/fineweb-2-edu-japanese
5) https://huggingface.co/datasets/fujiki/wiki40b_ja
6) https://huggingface.co/datasets/cl-nagoya/ruri-dataset-v2-pt
7) https://huggingface.co/datasets/cl-nagoya/ruri-v3-dataset-ft

表 1　JMTEB 総合結果（%）．列ごとの最高値を太字で表示．

Model	Avg	Retrieval	STS	Classification	Reranking	Clustering
Sentence-Sarashina-Bi-0.5B (Decoder-Pretrained)	**66.84**	**59.00**	**83.50**	**74.35**	**77.36**	49.40
Sentence-ModernBERT-JP-0.5B (Encoder)	65.31	57.95	80.78	71.73	75.50	**50.03**
Sentence-Llama-Bi-JP-0.5B (Decoder)	61.02	51.55	78.01	68.51	71.96	48.69

表 2　処理速度とメモリ効率の比較．NVIDIA H100 GPU を使用し，入力系列長 128 トークン，バッチサイズ 128 で実験．

Model	Throughput (samples/s)	Memory (MB)	Speedup vs. Decoder
Sentence-ModernBERT-JP-0.5B	**2205**	**2877**	**1.55**×
Sentence-Llama-Bi-JP-0.5B	1421	3885	1.00×

3.5　評価設定

構築したモデルの性能評価には，日本語文埋め込みベンチマークである JMTEB を用いる．JMTEB は，Retrieval, STS, Classification, Reranking, Clustering の 5 タスクで構成され，文埋め込みモデルを多面的に測定できる．また，運用時の処理速度とメモリ効率を評価するため， wiki40b_ja を用いて系列長 128 トークンの条件下でスループットと GPU メモリ使用量を測定する．この系列長は ModernBERT におけるローカルアテンションのサイズに対応しており，対称的に比較可能である．処理速度とメモリ効率の詳細な測定条件は付録 B に示す．さらに，埋め込み表現の品質評価として，Wikipedia[8] と MIRACL[9] を用いた Alignment と Uniformity の分析も行う．

4　実験結果

4.1　下流タスク性能

JMTEB による評価結果を表 1 に示す．

同一条件で構築した Encoder モデルと Decoder モデルを比較すると，平均スコアで Encoder モデルが Decoder モデルを 4.29pt 上回った．特に，Retrieval タスクでは 6.40pt の差が見られ，検索タスクにおいて Encoder モデルの優位性が顕著である．一方，大規模事前学習済みの Decoder モデルは，同規模のランダム初期化 Decoder モデルを 5.82pt 上回り，さらに同一条件で文埋め込み学習をした Encoder モデルも 1.53pt 上回った．これは，10T トークンの大規模事前学習により獲得した知識が文埋め込みタスクにも有効に転用できることを示している．

4.2　処理速度とメモリ効率

運用を想定した処理速度とメモリ効率の評価結果を表 2 に示す．

系列長 128 トークン，バッチサイズ 128 の条件下で測定した結果，Encoder モデルは 2205 samples/s のスループットを達成し，Decoder モデルの 1421 samples/s に比べて約 1.55 倍高速であった．GPU メモリ使用量については，Encoder モデルが 2877 MB であるのに対し，Decoder モデルは 3885 MB を消費している．この系列長 128 トークンは，ModernBERT におけるローカルアテンションのサイズに対応しており，公平な比較となっている．詳細な測定結果は付録 B に示す．

これらの結果から，RAG システムの運用において Encoder モデルが計算効率面で有利であり，大規模コーパスの高速な埋め込みに効果的であることがわかる．

4.3　埋め込み表現の品質

図 2 に，Alignment と Uniformity による埋め込み表現の品質評価結果を示す．

図 2 に示すように，同一条件 Decoder モデルが最も良好な Alignment と Uniformity を示し，次いで事前学習済み Decoder モデル，Encoder モデルの順となった．しかし，この順序が下流タスクの性能順序と一致していない．表 1 で示したように，下流タスクでは事前学習済み Decoder，Encoder，同一条件 Decoder の順となっている．この乖離は，Alignment と Uniformity が埋め込み表現の性質を測定する指標であり，下流タスクで要求される意味的な関係性を完全に捉えきれていない可能性を示唆している．

8）https://huggingface.co/datasets/wikimedia/wikipedia
9）https://huggingface.co/datasets/miracl/miracl

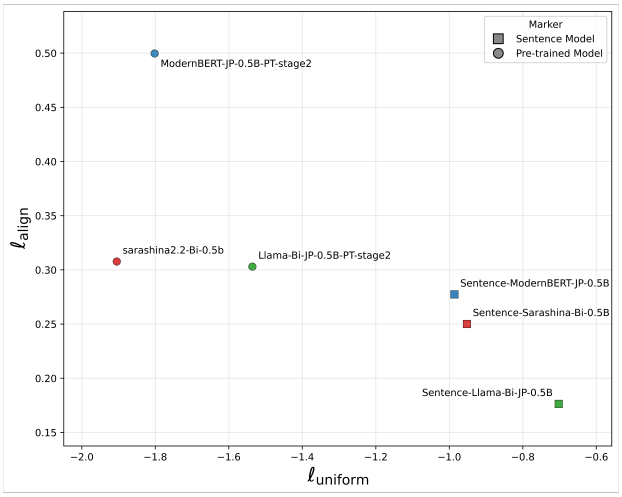

図 2　埋め込み学習後の Alignment と Uniformity．四角は文埋め込みを学習したモデル，丸は事前学習のみを行ったモデルを示す．右下ほど良好な表現品質を示す．

5　考察

5.1　アーキテクチャの性能差

同一条件下では Encoder モデルが下流タスク性能の面で Decoder モデルを上回った．この性能差の要因として，事前学習の目的関数の違いが挙げられる．Encoder モデルの Masked Language Modeling は文中の双方向的な依存関係を学習するのに対し，Decoder モデルの Causal Language Modeling は次単語予測に特化している．文埋め込みでは文全体の意味を単一ベクトルで表現する必要があり，双方向的な文脈理解が重要となる．

LLM2Vec では文埋め込み学習時に Attention を双方向化するが，事前学習で獲得された因果的な表現パターンは容易には変化しない．事前学習で学習された中間層の表現は次単語予測に最適化されており，文全体の意味的統合とは根本的に異なる．この目的関数の乖離が，Attention 機構の変更だけでは解消できず，性能差として現れたと考えられる．

5.2　表現品質指標の限界

Alignment と Uniformity による評価では，同一条件 Decoder モデルが最も良好な値を示したが，この順序は下流タスク性能と一致しなかった．これらは埋め込み空間の性質を測定する指標であり，類似サンプルの近接性や特徴分布の均一性を捉える．しかし，下流タスクで要求される意味的関連性や異なるクラス間の区別能力とは必ずしも一致しない．

5.3　大規模事前学習の効果

学習済み Decoder モデルが同一条件の Encoder モデルを 1.53pt 上回った要因として，事前学習データ量の差が挙げられる．同一条件モデルは 11B トークンで事前学習されているのに対し，学習済み Decoder モデルは約 10T トークン，すなわち約 1000 倍のデータ量で事前学習されている．この大規模コーパスで獲得された豊富な言語知識が，文埋め込みタスクに効果的に転用されたと考えられる．新規に大規模事前学習を行うには膨大な計算コストが必要となるが，既存の学習済みモデルを活用すれば，追加の文埋め込み学習のみで高性能なモデルを構築できる．文埋め込みの学習コストは事前学習に比べると低く，費用対効果の観点からも学習済みモデルの転用は有効な手法である．

5.4　実用的な指針

本研究の結果から，新規学習の場合は Encoder モデルを基本方針とし，学習済みモデルが利用可能な場合は Decoder モデルの転用を検討するという選択指針が得られた．新規にモデルを構築する場合は Encoder モデルが推奨される．同一の学習データ・コストでより高い性能を達成でき，推論時のスループットも約 1.55 倍優れている．コスト・処理速度を重視する場合に適している．一方，大規模事前学習済み Decoder モデルが利用可能な場合は，追加学習コストを抑えつつ高性能なモデルを実現でき，汎用的なタスクで最高精度を目指す場合には学習済み Decoder モデルの転用が推奨される．

6　おわりに

本研究では，Encoder と Decoder アーキテクチャの違いが文埋め込み性能に与える影響を明らかにするため，同一のパラメータ数・学習データ・学習手法で両モデルを構築し，性能・計算効率・表現品質を定量的に比較した．その結果，同一条件下では Encoder モデルが Decoder モデルを上回り，処理速度でも優位であることが示された．一方，大規模事前学習済み Decoder モデルの転用は，追加コストを抑えて性能向上をもたらす有効な手法であることを確認した．これらの知見は，実用的な文埋め込みモデルの選択指針を提供するだけでなく，アーキテクチャ選択が性能に与える影響の理解を深める重要な示唆となる．

参考文献

[1] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In **Proceedings of the 34th International Conference on Neural Information Processing Systems**, NIPS '20, 2020.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In **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, June 2019.

[3] Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddyi. Llm2vec: Large language models are secretly powerful text encoders. **Proceedings of the First Conference on Language Modeling**, 10 2024.

[4] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. **arXiv preprint arXiv:2302.13971**, 2023.

[5] Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Nv-embed: Improved techniques for training llms as generalist embedding models. **arXiv preprint arXiv:2405.17428**, 2024.

[6] Zheng Liu, Chaofan Li, Shitao Xiao, Yingxia Shao, and Defu Lian. Llama2Vec: Unsupervised adaptation of large language models for dense retrieval. In **Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**, pp. 3490–3500, August 2024.

[7] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. **arXiv preprint arXiv:2212.03533**, 2022.

[8] Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. Towards general text embeddings with multi-stage contrastive learning. **arXiv preprint arXiv:2308.03281**, 2023.

[9] Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. C-pack: Packed resources for general chinese embeddings. In **Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval**, SIGIR '24, p. 641–649, 2024.

[10] 塚越駿, 笹野遼平. Ruri: 日本語に特化した汎用テキスト埋め込みモデル. 言語処理学会第 31 回年次大会 (NLP2025), 2025.

[11] Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. MTEB: Massive text embedding benchmark. In **Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics**, pp. 2014–2037, May 2023.

[12] Shengzhe Li, Masaya Ohagi, and Ryokan Ri. Jmteb: Japanese massive text embedding benchmark, 2024. Accessed: 2025-10-11.

[13] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In **International conference on machine learning**, pp. 9929–9939. PMLR, 2020.

[14] Tianyu Gao, Xingcheng Yao, and Danqi Chen. SimCSE: Simple contrastive learning of sentence embeddings. In **Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing**, pp. 6894–6910, November 2021.

[15] Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Griffin Thomas Adams, Jeremy Howard, and Iacopo Poli. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. In **Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**, pp. 2526–2547, July 2025.

[16] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Thomas Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karén Simonyan, Erich Elsen, Oriol Vinyals, Jack Rae, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. In **Advances in Neural Information Processing Systems**, Vol. 35, pp. 30016–30030, 2022.

[17] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. **arXiv preprint arXiv:1807.03748**, 2018.

表 3　事前学習・WSL・FT のハイパーパラメータ．Ep はエポック数，BS はバッチサイズ，Len は入力系列長を示す．

段階	データセット	Ep	学習率	温度	BS*	Len
事前学習 1	hotchpotch/fineweb-2-edu-japanese (sample_10BT)	1	5×10^{-4}	–	16 (×32)	1024
事前学習 2	fujiki/wiki40b_ja	2	5×10^{-5}	–	2 (×32)	8192
弱教師あり (WSL)	cl-nagoya/ruri-dataset-v2-pt†	1	5×10^{-5}	0.01	128 (×4)	256
教師あり (FT)	cl-nagoya/ruri-v3-dataset-ft	1	5×10^{-6}	0.01	32 (×2)	512

* NVIDIA H100 GPU 8 枚環境での per-device BS と勾配蓄積回数．
† 各サブセットあたり 200 万件のペアを用いて弱教師あり学習を実施．

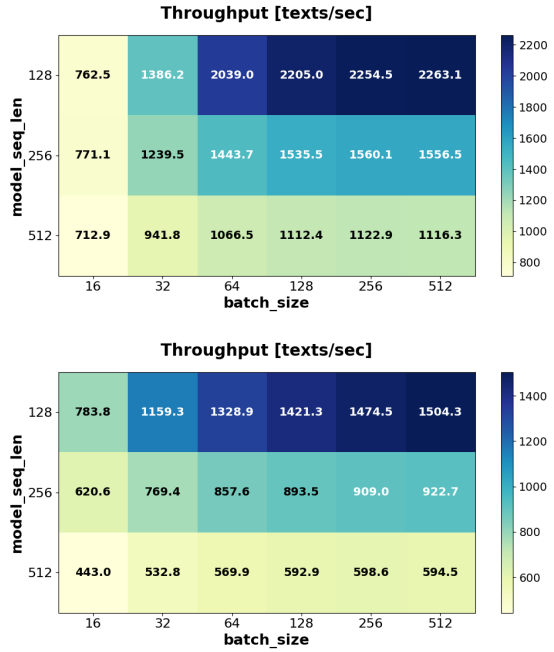

図 3　バッチサイズと系列長を変化させた際のスループット．上が Encoder，下が Decoder の結果を示す．

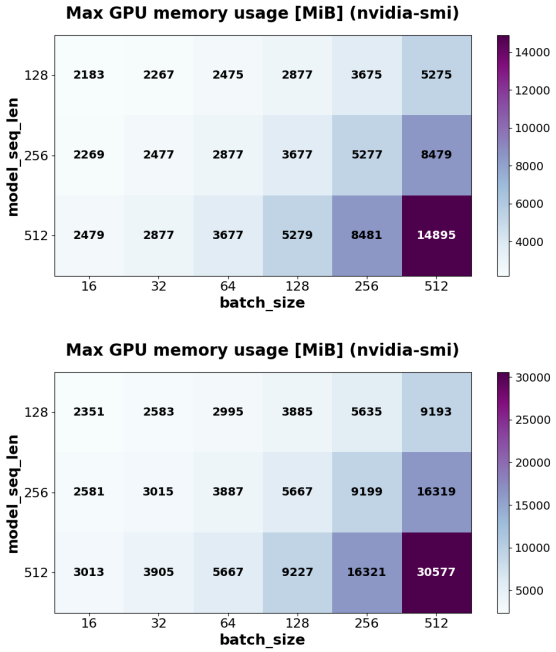

図 4　バッチサイズと系列長を変化させた際の GPU メモリ使用量．上が Encoder，下が Decoder の結果を示す．

A　学習設定の詳細

表 3 に事前学習・WSL・FT 各段階の設定を示す．対照学習には InfoNCE loss [17] を用いる．WSL，FT では，GTE に倣い task-homogeneous batching を導入し，同一バッチ内に同一データセットのサンプルのみを含めることで，バッチ内負例の識別難易度を適切に保ち，padding による計算コストも削減した．

B　詳細な効率性評価結果

B.1　測定条件と評価方針

本節では，系列長とバッチサイズを変化させた処理速度とメモリ効率の測定結果を示す．本文では系列長 128 トークンでの結果を報告したが，長い文書も想定し系列長 256・512 トークンの結果も示す．ただし，長い系列長では ModernBERT がローカルアテンションの制約により完全なアテンションを実現できないため，Encoder モデルに有利な比較となる点に留意されたい．また，バッチサイズについては 16 から 512 まで変化させ，評価した．

B.2　スループットおよびメモリ使用量

図 3 および図 4 に，バッチサイズと系列長を変化させた際のスループットと GPU メモリ使用量の測定結果を示す．

スループットについては，ほぼすべての条件において Encoder モデルが一貫して優位である．一方，メモリ使用量については，全条件において Encoder モデルが Decoder モデルより少ない傾向を示し，特に系列長 256・512 の条件下では両モデル間の差が顕著に開く結果となった．