複数ノードを用いた言語モデルの構築とドメイン適応

鈴木 雅弘[1,2,a)]　坂地 泰紀[3,b)]

概要：BERT や LLaMA などの言語モデルの構築は，GPU を利用した大規模な計算資源が必要である．本稿では，複数ノードを使用して BERT，DeBERTaV2 と LLaMA の事前学習やインストラクションチューニングを行うための環境の実例を紹介する．高速なネットワークや分散ファイルシステム，及びノードごとに異なるバッチサイズでの学習のための実装などの複数の方策により，より効率的な学習が可能となる．これらの知見は研究室内部のみに蓄積されることが多く，広く共有されることが少ない傾向にある．研究室単位で実施可能な言語モデルの構築およびチューニングのための施策やインフラ基盤について整理することで，本稿が有用な情報源となることが期待される．

Language Model Construction and Domain Adaptation using Multiple Nodes

1. はじめに

大規模言語モデル (Large Language Model, LLM) を始めとした自然言語処理技術の発展とともに，言語モデルは著しい性能向上が進んでいる．特に，BERT [1] や GPT シリーズ [2], [3], [4] を始めとした近年のモデルの多くが，Transformer 機構 [5] の事前学習によって高い性能を発揮している．これらを更に発展させたモデルとして，ChatGPT [6] や DeBERTaV3 [7]，Llama2 [8]，BLOOM [9] などが登場している．

これらのモデルは，年々サイズが大きくなっており，日本語で公開されるモデルも 70 億 (7B) 以上のパラメータを持つものが多い [*1]．例えば 7B のパラメータサイズを持つモデルの場合，デフォルトの設定で GPU で推論のみをする場合でも，約 28GB の GPU メモリが必要となる．これは一般的なコンシューマー向け GPU[*2]においては，学習だけでなく推論をすることも困難である．また GPU メモリの大きい NVIDIA A100 80GB を使用する場合においても，大きなバッチサイズで学習することは困難である．さらに，公開されているモデルの多くは一般的なドメインのタスクを解く性能は高いものの，金融や医療といった専門性の高いドメインでは，各ドメインのコーパスを用いて事前学習やインストラクションチューニング [10] を行うことで，より高い性能が得られることが報告されている [11], [12]．そのため，多くの GPU を持つスーパーコンピューターで学習された汎用モデルが公開されても，研究室などのオンプレミスの環境で継続事前学習やインストラクションチューニングを行うニーズは存在する．

一般的に，研究室で行われるモデルの構築やチューニングを行うための，特にネットワークやファイルシステムに関連した知見やノウハウは研究成果として公開されることは少なく，研究室内のみで蓄積される事が多い．しかし，これらの貴重な情報が外部に公開されず，他の研究者に共有されないことはあまり好ましくない．例えば，研究室内での知見が外部に公開されないことで，他の研究者が同じ問題に取り組む際に再び同じ課題に直面することがある．研究基盤となるネットワークやファイルシステムの知見が共有されることで，様々な研究を促進させることが期待される．

本稿では，大きな計算量が必要となる事前学習や大規模モデルのインストラクションチューニングに対し，研究室などのオンプレミスの計算資源で分散学習を行う場合の実例を紹介する．高速なネットワーク，分散ファイルシステ

[1] 東京大学
[2] 日興アセットマネジメント株式会社
[3] 北海道大学
[a)] research@msuzuki.me
[b)] sakaji@ist.hokudai.ac.jp
本稿の内容は筆者らが所属する組織を代表するものではなく，本稿の誤りは筆者らの責に属するものである．
[*1] https://github.com/llm-jp/awesome-japanese-llm/blob/main/README.md に公開モデルの一覧がまとめられている
[*2] NVIDIA RTX 4090 24GB を想定

表 1: 約 100GB の書き込みにおけるディスクの読み書き速度のベンチマーク．通信速度は理論値であり，実測値ではない．

端末	ディスク	通信速度	書き込み時間 (秒)	書き込み速度 (MB/秒)
オンプレミス	HDD	-	405	259
オンプレミス	SSD	-	145	723
オンプレミス	NVMe	-	90	1200
NFS	HDD	1Gbps	1063	99
NFS	HDD w/ NVMe キャッシュ	10Gbps	94	1100
NFS	SSD w/ NVMe キャッシュ	10Gbps	94	1100

ムやノードごとに異なるバッチサイズでの学習のための実装などの複数の方策により，より効率的な学習が可能となる．

2. 学習環境の整備

効率的に分散ノード上で事前学習やインストラクションチューニングを行うため，ハードウェア・ソフトウェアの両面から学習環境の整備を行う．ハードウェア面では，ネットワークの高速化とノード間で共有可能なファイルシステムの構築を行う．ソフトウェア面では，異なる GPU メモリ量を持つノードを効率的に利用するための実装や，実用的で簡単に利用できる施策を活用する．

2.1 通信・ファイル共有システム

通常の 1Gbps の通信環境に対し，より高速な 10Gbps を実現するためには，LAN ケーブル (RJ45) においてカテゴリー 6A 以上を使用するか，光ファイバーを利用することが多い．本稿の環境では，複数の部屋間を接続する必要性から，10Gbps より大きい速度を実現できる光ファイバーを採用した．10Gbps でネットワークを構築することで，BERT の base モデルを 4 ノードで構築した場合に，1Gbps に比べて約 3.5 倍の学習速度の向上が見られた．

複数のノードで分散学習環境を構築する場合に，事前学習で用いる大量のコーパスをそれぞれのノードにコピーして保存することは効率的とは言えない．そこで，ノード間で共有できる，アクセス速度と容量の異なる 3 つの Network File System (NFS) を構築する．1 つはアクセス速度が 1Gbps で HDD を用いた大容量のもの，2 つ目はアクセス速度が 50Gbps[*3]で NVMe のキャッシュと HDD を用いた大容量のもの，3 つ目はアクセス速度が 50Gbps[*3]で NVMe のキャッシュと SSD を用いた小容量のものである．

これらのファイル書き込みのパフォーマンスを比較するために，`dd if=/dev/zero of=zero1 bs=1MiB count=100000` の実行時間を `time` コマンドで計測した結果を表 1 に示す．最も書き込み速度が速かったのはオンプレミスで NVMe を使用するものであったが，NFS においても NVMe キャッシュを利用することでオンプレミスの NVMe よりやや遅い程度の速度が出ている．また，これらの書き込み速度は 1100 MB/秒であることから，10Gbps による恩恵を受けており，より高速なネットワークによる効果が見られた．NFS において NVMe キャッシュを利用した HDD と SSD の 2 種類では速度に差がなかった．今回の約 100GB の書き込みでは，NVMe のキャッシュ容量を超えず，SSD や HDD への直接の書き込みが計測時に発生しなかったためと考えられる．3 節で述べる事前学習で約 1TB のコーパスを用いた際には，HDD w/ NVMe キャッシュより SSD w/ NVMe キャッシュの方が高速であることを確認している．

[*3] 実際の利用では，光ファイバーのスイッチとケーブルによる 10Gbps の理論速度の上限があった

2.2 実験条件

表 2 に，事前学習モデルが提案された各論文で示されている，GPU に関連した計算資源の量を示す．一般的な研究室など，大規模な GPU サーバー群を持たない環境において，論文に記載されているバッチサイズをソフトウェア上の工夫なしにそのまま用いることは現実的ではない．以下では，小規模な GPU サーバー環境における事前学習やインストラクションチューニングを効率的に行うために利用できる手法を述べる．

2.2.1 異なるバッチサイズでの分散学習

既存の PyTorch[*4]の実装では，各ノードの GPU に対して同じバッチサイズで学習させることが前提とされている．そこで PyTorch の `DistributedSampler` クラスに関連した実装に編集を加え，ノード間で異なるバッチサイズで学習することを可能とした．具体的には，PyTorch で分散学習をする際に各ノードに送られるミニバッチへの管理は，データセットにおけるインデックスによって行われる．インデックスの配分を各ノードで指定したバッチサイズに応じて行うことで，ノード間で異なるバッチサイズを用いて学習が可能となった．これにより，32GB と 48GB の組み合わせのように，複数の GPU メモリ容量が混在する計算環境においても，それぞれの GPU メモリを最大限利用できるようになる．なお，各ノードにおいて使用する GPU の枚数や，1 ノード内の GPU ごとのバッチサイズは同じである必要がある．これらの実装は BERT や ELECTRA，

[*4] `https://github.com/pytorch/pytorch`

表 2: 事前学習モデルの構築に必要な計算資源．DGX-1, DGX-2 は NVIDIA Tesla V100 32GB をそれぞれ 8 枚，16 枚用いて構成される [*]．GPU 日は LLaMA と Llama2 は NVIDIA A100 80GB，それ以外のモデルは NVIDIA V100 32GB を基準とする．

モデル	GPU 構成	学習時間	GPU 日
BERT-base [1]	TPUv3 (V100 32 枚)[**]	4 日 (7 日)[**]	- (224)[**]
RoBERTa-large [13]	V100 1024 枚	1 日	1024
DeBERTa-base [14]	DGX-2 4 台	4 日	256
DeBERTa-large [14]	DGX-2 6 台	20 日	1920
LLaMA-7B [15]	A100 80GB	-	3434
LLaMA-65B [15]	A100 80GB	-	42598
Llama2-7B [8]	A100 80GB	-	7680
Llama2-65B [8]	A100 80GB	-	71680

[*] https://www.nvidia.com/ja-jp/data-center/dgx-2/
[**] https://alaginrc.nict.go.jp/nict-bert/index.html

RoBERTa の日本語での事前学習が可能な実装とともに GitHub にて公開している[*5]．

2.2.2 浮動小数点精度の変更

浮動小数点精度は，モデルの重みや勾配などの数値を扱う際の形式である．浮動小数点は，正負の符号を表す符号ビット，指数を表す指数ビット，有効数字を表す仮数ビットから構成される．PyTorch などで一般に用いられるのは単精度 (FP32) であり，これは合計 32 ビットを符号部，指数部，仮数部にそれぞれ 1, 8, 23 ビットに割り当てる．半精度 (FP16) は，合計 16 ビットを符号部，指数部，仮数部にそれぞれ 1, 5, 10 ビットに割り当てる．FP16 は数値を表すビット数を FP32 に比べ削減できるため，GPU の使用するメモリ量や計算速度を改善することができる．その一方で，指数部が FP32 より少なく表現可能な数値の範囲が狭くなるため，指数部の大きさに影響を受けやすいニューラルネットワークではオーバーフローなどの問題が発生しやすい．bfloat16 (BF16) はこの問題を解決するために，合計 16 ビットを用い，符号部に 1 ビット，仮数部に 7 ビット，指数部に 8 ビットを割り当てる[*6]．割り当てるビット数を FP32 と同じにすることで，FP32 と同じ範囲の数値の表現が可能となり，FP32 と同じハイパーパラメータを用いることができる [16]．より大規模なモデルが増えてきた近年では，量子化により符号部，仮数部にそれぞれ 1，7 ビットを割り当てた INT8 や，INT8 よりさらに少ないビット数を用いた，より効率的な学習が提案されている [17], [18]．一方で，NVIDIA が提供する GPU は世代ごとに対応可能な浮動小数点精度が異なり，最新の H100 や RTX 6000 Ada が含まれる Hopper 世代と Ada Lovelace 世代では FP32，FP16，BF16 と INT8 の全てが利用可能であるものの，Ampere 世代の A100 では FP32，FP16 と BF16 のみ，Volta 世代の V100 では FP32 と FP16 のみが利用できるなど，利用可能な浮動小数点精度が異なる点には注意が必要である．

2.2.3 勾配累積

勾配累積 (Gradient Accumulation) は，言語モデルを大きいバッチサイズで学習する場合に，小さなミニバッチに分割しミニバッチごとの勾配の和を用いてパラメータを更新する．これにより，小さい GPU メモリでも大きいバッチサイズの学習を可能にする．本稿で述べるモデル構築では，GPU のメモリサイズに対して利用可能な最大のマイクロバッチサイズを設定し，実現したいバッチサイズを全ノードの合計のマイクロバッチサイズで除すことで勾配累積の値 (回数) を設定する．

2.2.4 DeepSpeed

DeepSpeed [19] はマイクロソフト社によって公開されているソフトウェアで，大規模かつ高速な深層学習を実現するための様々な機能が含まれている．その中でも，研究室のオンプレミス環境で行える学習には，Zero Redundancy Optimizer (ZeRO) [20] が有効である．一般的な分散学習では，全ての GPU に対してオプティマイザーの状態，学習の勾配とモデルのパラメータを全て保持することが必要であるが，この場合他の GPU で使用するオプティマイザーの状態と勾配も保有することになる．そこで，ZeRO では 3 つのステージに分けて GPU メモリの効率化を行う．1 段階目 (ZeRO 1) では，オプティマイザーの状態を分割し各 GPU で使用する分のみ保持することで，GPU メモリの削減を行う．2 段階目 (ZeRO 2) では，オプティマイザーの状態に加え，学習の勾配についても分割を行う．3 段階目 (ZeRO 3) では 2 段階目に加え，モデルパラメータも分割するため，1GPU にモデルが収まらない場合には特に大きな効果を示す．本稿で述べるモデル構築では，パラメータ数が全て 1B 以下であったため，ZeRO 1 または Zero 2 を用いて学習を行った．基本的にはモデルが大きくなるにつれより高いステージの ZeRO を用いたほうが良いものの，分散するパラメータの量が増えるに連れネットワークへの負荷がかかるため，実際の学習を行う際には複数のステージ，またそれぞれのハイパーパラメータについての調整が必要である．

3. モデルの構築

本節では，2 節の環境や実験条件を用いて構築した，3 つのモデルについて述べる．1 つ目は金融文書を用いて事前学習や継続事前学習した BERT-base モデル [21]，2 つ目は汎用コーパスを用いて事前学習した DeBERTaV2-base モ

[*5] https://github.com/retarfi/language-pretraining
[*6] https://cloud.google.com/tpu/docs/bfloat16

表 3: モデル構築において使用した計算効率向上の手法.

モデル	ノード数	GPU 構成	精度	勾配累積	ZeRO
BERT-base	4	A6000, V100	FP32	なし	2
DeBERTaV2-base	3	V100	FP16	あり	1
LLaMA-13B-LoRA	1	A100	BF16	あり	2

デル[*7]，3 つ目は LLaMA-13B [15] に対して日本語インストラクションデータを適用してインストラクションチューニングを行った LLaMA-13B-LoRA モデル [22] である．これら 3 つのモデルを構築する際に用いた構成を表 3 に示す．

BERT-base モデルの構築では，BERT モデル [1] を日本語の金融ドメインに適応させた．その際，汎用コーパスと金融コーパスを用いてスクラッチから行う事前学習と [23]，汎用モデルに対して金融コーパスを用いて行う継続事前学習 [24], [25] という，金融ドメインの適応のため 2 つの学習方法を比較した．BERT-base モデルは事前学習のバッチサイズが 256 で入力長が 512 トークンと，近年の Llama2 [8] を始めとした学習と比べ 1 ステップに必要な GPU メモリ量が小さい[*8]ため，4 ノード 8GPU の本稿の環境では，勾配累積を用いずに学習を行うことが可能であった．金融 BERT-base モデルの構築では，スクラッチからの事前学習と継続事前学習の間に大きな性能差は見られなかった．

DeBERTa [14] は，Attention の入力を文脈と位置の 2 つに分ける (ほどく) Desentangled Attention により学習の安定性を高め，高い性能を示したモデルである．DeBERTa の後継モデルである DeBERTaV2 では，モデルやトークナイザーに対していくつかのアップデートがなされている[*9]．本稿で述べる DeBERTaV2-base モデルの構築では，形態素解析を用いずにトークン分割を行うモデルを構築した．固有表現抽出 (Named Entity Recognition, NER) など単語のアライメントが重要なタスクに対応するために，日本語の Encoder モデルの多くが事前に形態素解析器によって単語分割を行う．分類タスクにおいては，形態素解析器を用いないモデルでも性能劣化がなく [26], [27]，単語分割をしないことでトークンの分割単位を長くできる．これによりモデルに入力が可能な文字列を長くすることができるというメリットがある．事前学習のコーパスとしては日本語の CC-100 [28]，mC4 [29]，OSCAR [30] の 2301 バージョン，Wikipedia，Wikinews を用いた．これらのデータセットを合わせた容量は約 400GB であり，NFS を用いてこれらを読み込む際には，NVMe キャッシュを持つ HDD より NVMe キャッシュを持つ SSD の方が読み込み速度が速く，表 1 では表出しなかった差が見られた．また他の 2 つのモデルと異なり，当該モデルの構築においては DeepSpeed の Zero1 を用いた．これは，筆者らの検証において Zero2 より Zero1 の方が，わずかに GPU メモリの消費は大きいものの，総合的な学習速度では Zero1 の方が優れていたためである．上述の BERT-base の構築では検証を行っていないものの，BERT-base の構築においても Zero1 を用いたほうが学習速度が速かったと考えられる．

LLaMA-13B-LoRA モデルの構築では，英語で構築された LLaMA [15] に対して日本語インストラクションデータを用い，インストラクションチューニング [10] を行った [22]．LLaMA-13B は BERT-base や DeBERTaV2-base の約 110M パラメータの約 12 倍のパラメータサイズを持つ．そのため，BERT-base や DeBERTaV2-base の学習設定をそのまま適用することは困難である．そこで，Low-Rank Adaptation (LoRA) チューニング [31] を PEFT ライブラリ [32] によって実装し，更新するパラメータ数を減らして学習を行った．当該モデルの構築に使用した実装は GitHub[*10]にて公開している．また，構築したモデルを広く利用できるように，2023 年 4 月から 6 月頃にかけて Hugging Face Spaces にてモデルの出力を試せるフレームワークを公開した[*11]．その際の実装では FastChat[*12], 8bit 量子化のために bitsandbytes [17]，Web アプリケーションのために gradio [33] を用いた．Hugging Face Spaces にて使用したリソースは NVIDIA T4 small (15GB Memory, $0.60/時間) であった．

4. まとめ

本稿では，研究室を念頭にした複数ノードでの分散学習において，言語モデルの構築やチューニングのために利用可能ないくつかの施策を述べた．10Gbps の通信環境や NFS による分散ファイルシステムを用いることで，事前学習やモデルのチューニングが効率的になる．ファイルの読み書きの高速化により，一般的な深層学習においても効率性が増すことが期待される．また，ノード間で異なるバッチサイズでの学習のための実装を行い，研究室などにある GPU メモリの全てを学習にあてることができるようになった．さらに浮動小数点の変更や勾配累積，DeepSpeed Zero の活用により更に効率的な学習が可能となる．実際にこれらを用いて BERT，DeBERTaV2 や LLaMA の事前学習やチューニングを行い，得られた知見について紹介した．これらの方策が，言語モデルの事前学習やチューニングのための有用な情報となることが期待される．

[*7] https://huggingface.co/izumi-lab/deberta-v2-base-japanese
[*8] Llama2 ではバッチサイズが 1,000，入力長が 4,000 である
[*9] https://github.com/microsoft/DeBERTa/blob/master/README.md#whats-new-in-v2
[*10] https://github.com/retarfi/jallm
[*11] https://huggingface.co/spaces/izumi-lab/llama-13b-japanese-lora-v0-1ep，現在は休止中
[*12] https://github.com/lm-sys/FastChat

謝辞

本研究は JST さきがけ JPMJPR2267 の助成を受けたものです.

参考文献

[1] Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 4171–4186 (online), DOI: 10.18653/v1/N19-1423 (2019).

[2] Radford, A., Narasimhan, K., Salimans, T. and Sutskever, I.: Improving Language Understanding by Generative Pre-Training (2018).

[3] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. and Sutskever, I.: Language Models are Unsupervised Multitask Learners (2019).

[4] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A. et al.: Language Models are Few-Shot Learners, *Advances in Neural Information Processing Systems*, Vol. 33, pp. 1877–1901 (2020).

[5] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. and Polosukhin, I.: Attention Is All You Need, *Advances in Neural Information Processing Systems*, Vol. 30, pp. 5999–6009 (2017).

[6] OpenAI: ChatGPT, `https://openai.com/blog/chatgpt/` (2023).

[7] He, P., Gao, J. and Chen, W.: DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing, *International Conference on Learning Representations* (2023).

[8] Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S. et al.: Llama 2: Open Foundation and Fine-Tuned Chat Models (2023).

[9] Scao, T. L., Fan, A., Akiki, C., Pavlick, E., Ilić, S., Hesslow, D., Castagné, R., Luccioni, A. S., Yvon, F., Gallé, M. et al.: BLOOM: A 176B-Parameter Open-Access Multilingual Language Model (2022).

[10] Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Gray, A. et al.: Training language models to follow instructions with human feedback, *Advances in Neural Information Processing Systems* (2022).

[11] Singhal, K., Tu, T., Gottweis, J., Sayres, R., Wulczyn, E., Hou, L., Clark, K., Pfohl, S., Cole-Lewis, H., Neal, D. et al.: Towards Expert-Level Medical Question Answering with Large Language Models (2023).

[12] Xie, Q., Han, W., Zhang, X., Lai, Y., Peng, M., Lopez-Lira, A. and Huang, J.: PIXIU: A Comprehensive Benchmark, Instruction Dataset and Large Language Model for Finance, *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track* (2023).

[13] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V. and Allen, P. G.: RoBERTa: A Robustly Optimized BERT Pretraining Approach (2019).

[14] He, P., Liu, X., Gao, J. and Chen, W.: DeBERTa: Decoding-enhanced BERT with Disentangled Attention, *International Conference on Learning Representations* (2021).

[15] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F. et al.: LLaMA: Open and Efficient Foundation Language Models (2023).

[16] Kalamkar, D. D., Mudigere, D., Mellempudi, N., Das, D., Banerjee, K., Avancha, S., Vooturi, D. T., Jammalamadaka, N., Huang, J., Yuen, H., Yang, J., Park, J., Heinecke, A., Georganas, E., Srinivasan, S., Kundu, A., Smelyanskiy, M., Kaul, B. and Dubey, P.: A Study of BFLOAT16 for Deep Learning Training (2019).

[17] Dettmers, T., Lewis, M., Belkada, Y. and Zettlemoyer, L.: LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale (2022).

[18] Dettmers, T. and Zettlemoyer, L.: The case for 4-bit precision: k-bit Inference Scaling Laws (2022).

[19] Rasley, J., Rajbhandari, S., Ruwase, O. and He, Y.: DeepSpeed: System Optimizations Enable Training Deep Learning Models with Over 100 Billion Parameters, *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Association for Computing Machinery, pp. 3505–3506 (online), DOI: 10.1145/3394486.3406703 (2020).

[20] Rajbhandari, S., Rasley, J., Ruwase, O. and He, Y.: ZeRO: Memory Optimizations toward Training Trillion Parameter Models, *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–16 (2020).

[21] Suzuki, M., Sakaji, H., Hirano, M. and Izumi, K.: Constructing and analyzing domain-specific language model for financial text mining, *Information Processing & Management*, Vol. 60, No. 2, p. 103194 (online), DOI: 10.1016/j.ipm.2022.103194 (2023).

[22] Hirano, M., Suzuki, M. and Sakaji, H.: llm-japanese-dataset v0: Construction of Japanese Chat Dataset for Large Language Models and Its Methodology, *Advances in Networked-based Information Systems*, pp. 442–454 (2023).

[23] Liu, Z., Huang, D., Huang, K., Li, Z. and Zhao, J.: FinBERT: A Pre-trained Financial Language Representation Model for Financial Text Mining, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pp. 4513–4519 (online), DOI: 10.24963/ijcai.2020/622 (2020).

[24] Araci, D.: FinBERT: Financial Sentiment Analysis with Pre-trained Language Models (2019).

[25] Gururangan, S., Marasović, A., Swayamdipta, S., Lo, K., Beltagy, I., Downey, D. and Smith, N. A.: Don't Stop Pretraining: Adapt Language Models to Domains and Tasks, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8342–8360 (online), DOI: 10.18653/v1/2020.acl-main.740 (2020).

[26] 鈴木雅弘, 坂地泰紀, 和泉 潔: 異なる単語分割システムによる日本語事前学習言語モデルの性能評価, 言語処理学会 第 29 回年次大会 (NLP2023), pp. 894–898 (2023).

[27] Fujii, T., Shibata, K., Yamaguchi, A., Morishita, T. and Sogawa, Y.: How do different tokenizers perform on downstream tasks in scriptio continua languages?: A case study in Japanese, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pp. 39–49 (online), DOI: 10.18653/v1/2023.acl-srw.5 (2023).

[28] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang,

S., Matena, M., Zhou, Y., Li, W. and Liu, P. J.: 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).

[29] Wenzek, G., Lachaux, M.-A., Conneau, A., Chaudhary, V., Guzmán, F., Joulin, A. and Grave, E.: CCNet: Extracting High Quality Monolingual Datasets from Web Crawl Data, *Proceedings of the 12th Language Resources and Evaluation Conference*, pp. 4003–4012 (online), available from ⟨https://www.aclweb.org/anthology/2020.lrec-1.494⟩ (2020).

[30] Ortiz Su'arez, P. J., Sagot, B. and Romary, L.: Asynchronous pipelines for processing huge corpora on medium to low resource infrastructures, Proceedings of the Workshop on Challenges in the Management of Large Corpora (CMLC-7) 2019. Cardiff, 22nd July 2019, pp. 9 – 16 (online), DOI: 10.14618/ids-pub-9021 (2019).

[31] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L. and Chen, W.: LoRA: Low-Rank Adaptation of Large Language Models, *International Conference on Learning Representations*, pp. 1–13 (2022).

[32] Mangrulkar, S., Gugger, S., Debut, L., Belkada, Y. and Paul, S.: PEFT: State-of-the-art Parameter-Efficient Fine-Tuning methods, https://github.com/huggingface/peft (2022).

[33] Abid, A., Abdalla, A., Abid, A., Khan, D., Alfozan, A. and Zou, J.: Gradio: Hassle-Free Sharing and Testing of ML Models in the Wild (2019).