Swallow コーパス: 日本語大規模ウェブコーパス

岡崎 直観 [1]　服部 翔 [1]　平井 翔太 [1]　飯田 大貴 [1,2]　大井 聖也 [1]
藤井 一喜 [1]　中村 泰士 [1]　Mengsay Loem [1]　横田 理央 [1]　水木 栄 [1,3]
[1] 東京工業大学情報理工学院　[2] 株式会社レトリバ　[3] 株式会社ホットリンク
{okazaki@c, kakeru.hattori@nlp.c, shota.hirai@nlp.c, taiki.iida@nlp.c,
masanari.ohi@nlp.c, kazuki.fujii@rio.gsic, taishi.nakamura@rio.gsic,
mengsay.loem@nlp.c, rioyokota@gsic, sakae.mizuki@nlp.c} .titech.ac.jp

概要

これまで、オープンな日本語大規模言語モデルの学習には CC-100、mC4、OSCAR などのコーパスの日本語部分が用いられてきた。ところが、これらのコーパスは日本語テキストの品質を重視して作られた訳ではない。本研究では Common Crawl のアーカイブ（2020 年から 2023 年にかけて収集された 21 スナップショット分、約 634 億ページ）から日本語のテキストを独自に抽出・精錬し、約 3,121 億文字（約 1.73 億ページ）からなる日本語ウェブコーパスを構築した。この規模は、CC-100 (約 258 億文字)、mC4（約 2,397 億文字)、OSCAR 23.10（約 740 億文字）を抜き、日本語の言語モデルの学習コーパスの中で、商用利用が可能なものとしては最大である。

1　はじめに

2022 年暮れに OpenAI が公開した ChatGPT は、汎用的な人工知能のマイルストーンとして世界に大きな痕跡を残した。自然言語処理や人工知能の研究開発の底上げ、LLM の知能のメカニズムの解明、海外の一握りの企業に依存してしまう安全保障上のリスク、信頼できる人工知能の実現など、動機は様々であるが、2023 年は日本語に強い LLM の開発・発表が相次いだ。ところが、日本語の LLM の学習に用いるコーパス（学習データ）は海外で開発されたものを採用する場合が多く、日本語に特化し、透明性の高いコーパスが利用されているとは言い難い。

表 1 に LLM の学習に用いられる代表的なコーパスを挙げた。日本語のテキストを含み、かつ学習後の言語モデルの利用に制限が付かないコーパスは、CC-100、mC4、OSCAR 23.01 が有力候補となるが、Common Crawl の WET 形式のテキストを処理したものであるため、HTML からテキストへの変換にお

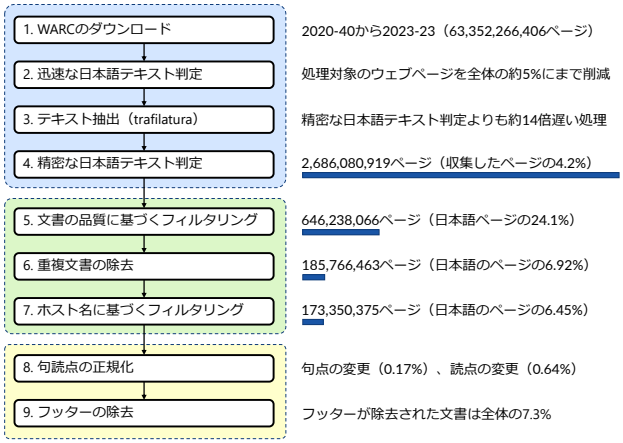

図 1　Swallow コーパスの構築手順

いてノイズが含まれることがある。また、これらは多言語コーパスとして構築されており、日本語は数多くある言語の一つであるため、日本語に特化してテキストの品質を高めるような工夫は取り入れられていない。さらに、日本語の LLM は評価データや評価指標の確立途上にあるため、コーパスの開発も暗中模索の状況にある。そこで、本研究では日本語の LLM の学習に活用されることを念頭に、大規模かつ品質の高い日本語のウェブコーパスを構築することを目指す。本研究で開発された **Swallow コーパス**は、Common Crawl の 21 件のスナップショット（CC-MAIN-2020-40 から CC-MAIN-2023-23 まで）から作られており、クリーニング後のコーパスの規模は 173,350,375 ページ、312,093,428,689 文字である。本研究で構築されたコーパスで Llama 2 の継続学習を行ったモデルを公開している[1)]。

2　手法

Swallow コーパスの構築手順を図 1 に示した。構築手順はおおよそ以下の 3 段階に分かれている。

1)　https://tokyotech-llm.github.io/swallow-llama

コーパス	英語の規模	日本語の規模	ライセンス	言語判定	テキスト抽出
CC-100 [1]	532 BT	26 BL	CC	fastText LID	WET
C4 [2]	156 BT	—	ODC-BY, CC	`langdetect`	WET
mC4 [3]	2,733 BT	240 BL	ODC-BY, CC	`CLD3`	WET
OSCAR 23.01 [4]	523 BW	74 BL	CC0	fastText LID	WET
Pile-CC [5]	2,312 BL	—	MIT License	`CLD2`	`jusText`
ROOTS [6]	484 BB	—		fastText LID	自家製
RefinedWeb [7]	600 BT	—	ODC-BY, CC	自家製	`trafilatura`
Dolma [8]	2,415 BT	—	ImpACT license	fastText LID	WET?
ClueWeb22 [9]	*16.7 TT	3,301 BL	非商用（要契約）	`BlingFire`	商用
RedPajama-Data-v2 [10]	20.5 TT	—	Apache	fastText LID	WET
Swallow（クリーニング前）	—	3,684 BL	（非公開）	自家製	`trafilatura`
Swallow（クリーニング後）	—	312 BL	（検討中）	自家製	`trafilatura`

表 1　大規模言語モデルの事前学習に用いられる代表的なコーパス。BL、BW、BT、TT はそれぞれ、billion letter（10 億文字）、billion word（10 億単語）、billion token（10 億トークン）、trillion token（1 兆トークン）の略である。英語のコーパスの規模は論文等での公表値をそのまま採用した。ClueWeb22 の英語の規模は元論文 [9] に基づくが、英語以外の言語の数も混ざっていると想定されるため、実際はこの 40%程度と思われる。日本語のコーパスの規模は、本研究でユニコードの文字数を実測した。テキスト抽出の WET は Common Crawl が配布しているテキスト抽出結果（WET 形式）を表す。

1. Common Crawl の WARC ファイルから日本語テキストを抽出する（2.1 節）
2. 品質フィルタリングおよび重複除去で日本語テキストを厳選する（2.2 節、2.3 節、2.4 節）
3. テキスト内の正規化を行う（付録 C 節）

本構築手順は RefinedWeb [7] を参考に、処理を日本語向けにカスタマイズした。

2.1　日本語のテキスト抽出

Common Crawl のスナップショットは Amazon S3 のバケットとして保管されており、S3 もしくはウェブサーバ[2]経由でアクセスできる[3]。本研究では `WARCIO`[4] ライブラリを用いて、WARC 形式のデータから HTML コンテンツを抽出した。Common Crawl 全体の約 5%程度が日本語で書かれたウェブページと見積もられる[5]ため、うまく日本語のウェブページだけを選別してコーパス構築処理を実行できれば、処理時間を約 95%節約できる。ところが、言語判定の精度を高めるためには、HTML のマークアップを取り除いてから言語判定器にかけることが望ましく、テキスト抽出を先に適用してから言語判定を行えばよいが、言語判定よりもテキスト抽出の方が処理時間を要するので、この順番でコーパス構築を行ってしまうと、処理時間を削減できない。

そこで、図 1 の「2. 迅速な日本語テキスト判定」では、テキスト抽出を行わず迅速な方法で日本語である可能性が高いウェブページを選別する（詳細は付録 A 参照）。この手順で候補として選ばれたウェブページに対して、`Trafilatura`[6]で「3. テキスト抽出」を行い、「4. 精密な日本語テキスト判定」で言語判定を精密に行う（詳細は付録 B 参照）。インテル Xeon Gold 6130 プロセッサ（2.1GHz）の 1 CPU を用いたベンチマーク実験では、迅速な日本語テキスト判定により、WARC ファイルから日本語テキストを取り出す処理が約 15 倍高速化された。また、`whatlang-corpora`[7]（6,718,883 件）を用いた評価実験では、精密な日本語テキスト判定の適合率は 0.999、再現率は 0.979、F1 スコアは 0.989 であった。

本節で説明した日本語テキストの抽出処理により、2,686,080,919 ページ、3,684,320,219,695 文字の日本語テキストが得られた。

2.2　品質フィルタリング

この処理では、(1) 繰り返しの多いウェブページの除去、(2) 品質のよい日本語の文章を含むウェブページの選別、(3) 有害な表現を含むと思われるウェブページの除去、を行う。(1) の処理として、本研究では Rae ら [11] のルールをそのまま採用し、重複した表現が多い文書を削除した[8]。

2）https://data.commoncrawl.org/
3）https://commoncrawl.org/get-started
4）https://github.com/webrecorder/warcio
5）https://commoncrawl.github.io/cc-crawl-statistics/
6）https://trafilatura.readthedocs.io/
7）https://github.com/whatlang/whatlang-corpora
8）*n*-gram は文字単位で取る（形態素解析は行わない）。

続いて、(2) の日本語としての品質については独自にルールを設計し、次のいずれかを満たす文書は品質が低いと判断して、削除した（括弧内は条件）。

- 文字数 (400 文字未満)
- 平仮名の文字の割合 (0.2 未満)
- カタカナの文字の割合 (0.5 以上)
- 日本語の文字 (平仮名、カタカナ、漢字、句読点）の割合 (0.5 未満)
- 文書中の文の文字数の平均 (20 未満、もしくは 90 よりも多い場合)
- 最も長い文の文字数 (200 文字以上)
- 文末が省略記号で終わる文の割合 (0.2 以上)

なお、これらのルールは削除される／残される文書を目視で確認しながら、著者らが人手で調整した。(3) の有害な表現を含むと思われるウェブページの除去では、NG 表現を含む割合が文字数単位で 0.05 以上になる場合、そのテキストを削除した。

この品質フィルタリングにより、コーパスが 646,238,066 ページ（1,202,868,044,631 文字）に絞り込まれた。フィルタリング前後のテキストを目視で比較すると、EC サイトなど、商品名やキーワードが繰り返されているページが削除され、見た目にもテキストの品質が良くなったことを実感できる。

2.3 重複除去

Common Crawl は同一のウェブサイトを異なる時期に巡回・収集しているため、同一のウェブサイト上で加えられた軽微な修正や、転載などで内容が類似したウェブサイトを含む。現段階では、LLM を学習するときはコーパスの丸覚えを防ぐため、同じテキストでモデルの学習を繰り返すことは避けるべきと言われている。そこで、LLM の事前学習用のコーパスでは、重複したテキストを除去する処理、すなわち重複除去（deduplication）が行われる。ところが、億を超える数の文書の組に対して、ジャッカード係数やコサイン係数でテキストの類似度を総当たりで計算し、類似度の高いテキストを除去するという処理は膨大な時間を要する。

そこで、本研究では Lee らの研究 [12] を参考にして、MinHash [13] による重複除去を行った。MinHash を重複検出に用いるときは、MinHash 値を b 個連結したもの（バケットと呼ばれる）を r 個作成しておき、2 つの文書のバケットの組を r 回比較し、その中で一組でも完全一致する場合、2 つの文書は重複であると見なすアルゴリズムを採用することが一般的である。本研究では、$b = r = 20$ とし、$K = br = 400$ 個のハッシュ値を用いる設定を採用し、ジャッカード係数が 0.9 である文書の重複を約 92.5%の確率で検出できるように設定した[9)]。

本研究では、内容に重複があるウェブサイトの組がある場合、クローリングされた日時が新しいものを残し、古い方を削除した。今回実験に利用したスナップショットでは、2023 年 3 月から 6 月にかけて収集されたウェブページの残存率（重複除去で削除されない割合）は 77.8〜87.9%であるのに対し、2023 年 2 月以前に収集されたウェブページの残存率は 40%未満に低下し、2020 年頃に収集されたウェブページの残存率は 20%を下回った。この重複文書の除去処理により、コーパスの量は 185,766,463 ページ（353,191,872,805 文字）に減少した。

2.4 ホスト名に基づくフィルタリング

2.2 節において、ページ毎に NG 表現を含む割合を測定し、閾値を超えたものを削除していたが、より安全側に倒すためにホスト名の NG リストを作成し、ホスト名による削除を行った。以下の基準を満たすホスト名からなるブロックリストを作成した。

1. UT1 blocklist[10)] に収録されている
2. 出会い系サイトのサービス名を一度でも含むページの割合が 0.001 を超える場合
3. NG 表現を含む割合（文字数による割合）が 0.005 を超える場合[11)]。
4. `*wikipedia.org`
5. `*.5ch.net`

ブロックリストの作成では網羅性を重視し、適合率を犠牲にしているため、有害ではないウェブサイトもブロックリストに登録されることがある[12)]。したがって、この処理でブロックリストに登録されたホストが有害であるとは限らない。このフィルタリング処理で、コーパスの量は 173,350,375 ページ（312,674,522,606 文字）となった。

9） 本研究では、文字の 5-gram で特徴量を構成した。
10） https://dsi.ut-capitole.fr/blacklists/
11） ただし、文脈によっては有害ではない表現（「児童」など）や、別の単語の一部として使われる文字列（「ピエロ」など）があるため、ホワイトリストの表現の部分文字列は割合の計算から除外した。
12） 例えば、あるホストの特定のディレクトリやユーザのみが有害な表現を使っている場合や、有害ではない文脈で NG 表現を使っているテキストを含むホストは、ブロックリストに登録されてしまう恐れがある。

コーパス	QA JComQA	QA JEMHopQA	QA NIILC	機械読解 JSQuAD	自動要約 XL-Sum	日英翻訳 WMT20	英日翻訳 WMT20	数学 MGSM	平均
ベースモデル	0.6819	0.4455	0.4174	0.8551	0.2133	0.1981	0.2136	0.1320	0.3946
Swallow（本研究）	0.7837	0.5063	0.6406	0.9007	0.2168	0.1771	0.2737	0.2160	0.4644
ClueWeb22	0.7676	0.4929	0.5602	0.8955	0.2015	0.2332	0.2908	0.1160	0.4447
llm-jp-corpus v1.0.1	0.7471	0.5085	0.6034	0.9028	0.2191	0.1789	0.2589	0.1800	0.4498

表 2　各コーパスで継続事前学習を行ったモデルの日本語ベンチマークデータでの評価結果

3　実験

本研究で構築した日本語ウェブコーパスを用いて、日本語を強化した LLM を構築する実験を行った。今回は、Meta 社が公開している Llama 2 13B (base) モデル[13]に対して、日本語のコーパス等で継続事前学習を行った。LLM の学習には、Megatron-LM[14]を採用した。日本語の学習コーパスの違いによる LLM の性能の差を検証するため、ClueWeb22 [9] と llm-jp-corpus v1.0.1[15]でも継続事前学習を行った。継続学習のコーパスには日本語 Wikipedia、RefinedWeb、The Pile (arXiv) を混ぜた。正確には、4,096 トークンの系列長かつ 1,024 のバッチサイズで 25,000 のステップ数で継続事前学習をすることを想定し、約 104.9 BT の学習データを準備した。日本語と英語のトークンの比率を 9:1 とし、学習データの 5%を RefinedWeb の英語テキスト、5%を The Pile の arXiv 論文テキスト（英語）とし、残りの 90%を日本語のテキストとした。その 90%の日本語テキストの内訳は、日本語 Wikipedia が約 1.6 BT を占め、残りを日本語のウェブコーパスで埋めた。

3.1　評価データセット

本研究では，評価ベンチマークとして llm-jp-eval [14] と lm-evaluation-harness[16]を用いた。llm-jp-eval は多値選択式質問応答 (MC: multiple choice)、自由記述式質問応答 (QA: queston answering)、機械読解 (RC: reading complehension)、自然言語推論 (NLI: natural language inference) のタスクからなるベンチマークである。MC は JCommonsenseQA [15]、QA は JEMHopQA [16] と NIILC [17]、RC は JSQuAD [15] が用いられている。NLI については、言語モデルの推定ラベルが偏る傾向があり、その偏ったラベルが正解ラベルに偶然一致する場合にスコアが高くなる傾向が見られたため、今回の実験結果から除外することにした。lm-evaluation-harness は、Stability AI が開発したものの一部を用いた。今回は、自動要約タスクとして XL-Sum [18]、算術推論タスクとして MGSM [19] のそれぞれの日本語サブセットを用いた。さらに、日英・英日機械翻訳として WMT 2020 [20] を用いた。

3.2　評価結果

表 2 に評価結果を載せた。「(ベースモデル)」は継続事前学習を行わず、公開されている Llama 2 13B (base) をそのまま評価した時の結果である。いずれのコーパスを継続事前学習に用いても、全データセット平均で約 0.05〜0.07 ポイントのスコアの上昇が見られることから、継続事前学習の効果が確認できた。全データセットのスコアの平均をとると、本研究のコーパスで継続追加学習を行ったモデルの性能が最も高く、llm-jp-corpus v1.0.1 と ClueWeb22 が同程度のスコアで後に続く結果であった。

4　おわりに

本論文では、Common Crawl から日本語のテキストを独自に抽出・精錬し、約 3,121 億文字（約 1.73 億ページ）からなる日本語ウェブコーパス Swallow を構築した。構築したコーパスの品質を確認するため、Llama 2 13B の継続事前学習を行ったところ、既存のコーパスを用いた場合と比べて同等かそれを上回る性能の LLM を構築できた。

今後の課題は、有害表現や差別などの LLM の安全性に関する取り組みが挙げられる。現状では、単語リストやホスト名リストによる対応を行っているが、より信頼性の高いフィルタリング手法を確立し、有害なテキストを確実に除去しつつ、コーパスの規模を大きくしたい。また、本研究では継続事前学習を行ったが、今後は日本語の LLM をフルスクラッチで学習し、コーパスを評価したい。構築したコーパスで学習したモデルを質問応答や要約などの下流タスクで評価したが、これで LLM の「地頭の良さ」を測定できるのか疑問が残るので、事前学習コーパス単体での評価方法を検討していきたい。

13）https://huggingface.co/meta-llama/Llama-2-13b-hf
14）https://github.com/NVIDIA/Megatron-LM
15）https://github.com/llm-jp/llm-jp-corpus
16）https://github.com/Stability-AI/lm-evaluation-harness/tree/jp-stable

謝辞

この成果は、国立研究開発法人新エネルギー・産業技術総合開発機構（NEDO）の委託業務（JPNP18002）の結果得られたものです。また、LLM の継続事前学習の実験では、国立研究開発法人産業技術総合研究所が構築・運用する AI 橋渡しクラウド（ABCI: AI Bridging Cloud Infrastructure）の「大規模言語モデル構築支援プログラム」の支援を受けました。学習した LLM の評価実験では、LLM-jp（LLM 勉強会）で開発されているデータや公開されている知見を活用しました。

参考文献

[1] Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. CCNet: Extracting high quality monolingual datasets from web crawl data. In *Proc. of LREC*, pp. 4003–4012, 2020.

[2] 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.

[3] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mT5: A massively multilingual pre-trained text-to-text Transformer. In *Proc. of NAACL-HLT*, pp. 483–498, 2021.

[4] Julien Abadji, Pedro Ortiz Suarez, Laurent Romary, and Benoît Sagot. Towards a cleaner document-oriented multilingual crawled corpus. In *Proc. of LREC*, pp. 4344–4355, 2022.

[5] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800GB dataset of diverse text for language modeling. arXiv:2101.00027, 2020.

[6] Hugo Laurençon, Lucile Saulnier, Thomas Wang, Christopher Akiki, Albert Villanova del Moral, Teven Le Scao, et al. The BigScience ROOTS corpus: A 1.6TB composite multilingual dataset. arXiv:2303.03915, 2023.

[7] Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. The RefinedWeb dataset for Falcon LLM: Outperforming curated corpora with web data, and web data only. arXiv:2306.01116, 2023.

[8] Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Khyathi Chandu, Jennifer Dumas, Li Lucy, and Xinxi Lyu *et al.* Dolma: An open corpus of 3 trillion tokens for language model pretraining research. Technical report, Allen Institute for AI, 2023.

[9] Arnold Overwijk, Chenyan Xiong, Xiao Liu, Cameron VandenBerg, and Jamie Callan. ClueWeb22: 10 billion web documents with visual and semantic information. arXiv:2211.15848, 2022.

[10] together.ai. Redpajama-data-v2: An open dataset with 30 trillion tokens for training large language models. https://www.together.ai/blog/redpajama-data-v2, 2023.

[11] Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, and Susannah Young *et al.* Scaling language models: Methods, analysis & insights from training Gopher. arXiv:2112.11446, 2021.

[12] Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating training data makes language models better. In *Proc. of ACL*, pp. 8424–8445, 2022.

[13] Andrei Z Broder. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences 1997*, pp. 21–29, 1997.

[14] Namgi Han, 植田暢大, 大嶽匡俊, 勝又智, 鎌田啓輔, 清丸寛一, 児玉貴志, 菅原朔, Bowen Chen, 松田寛, 宮尾祐介, 村脇有吾, 劉弘毅. llm-jp-eval: 日本語大規模言語モデルの自動評価ツール. 言語処理学会第 30 回年次大会 (NLP2024), 2024.

[15] Kentaro Kurihara, Daisuke Kawahara, and Tomohide Shibata. JGLUE: Japanese general language understanding evaluation. In *Proc. of LREC*, pp. 2957–2966, 2022.

[16] 石井愛, 井之上直也, 関根聡. 根拠を説明可能な質問応答システムのための日本語マルチホップ QA データセット構築. 言語処理学会第 29 回年次大会 (NLP2023), pp. 2088–2093, 2023.

[17] 関根聡. 百科事典を対象とした質問応答システムの開発. 言語処理学会第 9 回年次大会 (NLP2003), pp. 637–640, 2003.

[18] Tahmid Hasan, Abhik Bhattacharjee, Md. Saiful Islam, Kazi Mubasshir, Yuan-Fang Li, Yong-Bin Kang, M. Sohel Rahman, and Rifat Shahriyar. XL-sum: Large-scale multilingual abstractive summarization for 44 languages. In *Findings of ACL-IJCNLP 2021*, pp. 4693–4703, 2021.

[19] Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. Language models are multilingual chain-of-thought reasoners. In *Proc. of ICLR*, 2023.

[20] Loïc Barrault, Magdalena Biesialska, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Matthias Huck, and Eric Joanis *et al.* Findings of the 2020 conference on machine translation (WMT20). In *Proceedings of the Fifth Conference on Machine Translation*, pp. 1–55, Online, 2020. Association for Computational Linguistics.

[21] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv:1412.6980, 2014.

[22] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Proc. of NeurIPS*, Vol. 35, pp. 16344–16359, 2022.

A　迅速な日本語テキスト判定

本研究では、以下の二つの条件のいずれかが満たされる場合のみ、テキスト抽出および言語判定を行った。

1. `<html lang="ja">` のように、日本語の HTML コンテンツであることが明示されていること
2. `<title>` タグの中身に対して精密な日本語テキスト判定器（B 節）を適用し、日本語と判定されること

このルールの妥当性を評価するため、後段の精密な日本語テキスト判定と迅速な日本語テキスト判定の結果を比較した。具体的には、精密な言語判定の結果が正しいと仮定し、`CC-MAIN-2023-23` の 10 個の WARC ファイル[17]（全 394,192 ページ、gzip 圧縮で計 12,052,992,707 バイト）を用い、迅速な言語判定の精度を測定したところ、適合率は 0.888、再現率は 0.967、F1 スコアは 0.926 であった。この結果から、迅速な日本語判定は日本語のウェブサイトの約 3.3%を捨ててしまう可能性があるが、適合率が比較的高いことから日本語以外の言語のウェブページのテキスト抽出の処理時間を削減できる。

B　精密な日本語テキスト判定

本研究では文字 n-gram を特徴量に用いた線形識別器で言語判定器を構築した。線形識別器の学習データとして、多言語の Wikipedia テキストを採用し、学習データから以下の基準のいずれかを満たす文字ユニグラム、バイグラム、トリグラムで特徴空間を構成した。

1. 全言語の学習データで出現頻度が上位 400,000 件以内
2. 日本語の学習データで出現頻度が上位 400,000 件以内
3. 中国語の学習データで出現頻度が上位 100,000 件以内
4. 各言語の学習データで出現頻度が上位 10,000 件以内

基準 (1) だけを用いると、日本語の文字に関する特徴量が十分に得られない可能性があるため、基準 (2) で日本語に特化した特徴量の獲得を狙った。基準 (3) は日本語と漢字を共有している中国語との区別をつけやすくすることを狙った。基準 (4) は、学習データ量が十分ではない言語のテキストに対する判定結果を安定化させる狙いがあった。特徴量の異なり数は 821,484 である。

言語判定器の学習では、Wikipedia CirrusSearch[18] の 2023 年 6 月 12 日版のデータを用いた。言語判定器の学習では、ダウンロードしたデータの 1/20 の量を用い[19]、残ったデータから開発データと評価データを 100,000 件ずつサンプリングした。線形識別器の学習には、`LIBLINEAR`[20] 2.46 で L2 正則化 L2 損失サポートベクトルマシンを学習し、開発データ上で言語判定精度が最も高くなる正則化係数を探索し、$C = 10$ に設定した。

C　テキスト内の正規化

構築したコーパスの句読点を「、。」に統一するため、以下のルールに基づいて句読点の「,.」を「、。」に置換した。

1. 記号「、」および「,」の文書内の出現回数を調べ、「,」の方が多く出現する場合は、「,」を「、」に置換する。ただし、直前が英数字である「,」は「、」への置換の対象から外す。
2. 記号「。」および「.」の文書内の出現回数を調べ、「.」の方が多く出現する場合は、「.」を「。」に置換する。ただし、直前が英数字である「.」は「。」への置換の対象から外す。

「。」と「.」、「、」と「,」のどちらの句読点が使われるかは、文書内で一貫していると仮定し、句読点の置換を行うかどうか判断することで、誤って置換してしまうケースを削減している。また、句点と読点、それぞれ独立に出現回数を比較することで、「、.」や「,。」などの句読点の正規化に対処している。この句読点の正規化処理により、290,318 文書（0.17%）の句点が「.」から「。」に、1,107,319 文書（0.64%）の読点が「,」から「、」に変更された。

なお、テキスト抽出に用いている `Trafilatura` は、ウェブページのナビゲーションやフッターのテキストを除去してくれるが、コーパス中で除去しきれなかったフッターテキストを見かけることがある。そこで、テキストの末尾 3 行に対して、「この記事へのトラックバック一覧」や「無断転載を禁ず」「クリック」などの表現を文字単位で 30%以上含む場合は、テキストから削除した。このフッターの除去処理により、12,617,787 文書（7.3%）のテキストの末尾にあるフッターが除去された。

D　継続事前学習の詳細

D.1　学習環境

継続事前学習では、産総研の AI 橋渡しクラウド (ABCI) を利用した。混合精度（`bfloat16`）を採用し、NVIDIA A100 ノードを複数台使用し、分散並列学習を行った。各ノードは NVIDIA A100 40GB GPU を 8 基を搭載し、ノード間は InfiniBand HDR にて接続されている。

D.2　分散学習手法

効率的な学習を行うためにデータ並列（data parallelism）、テンソル並列化（tensor parallelism）、パイプライン並列化（pipeline parallelism）を統合した 3D 並列化（3D parallelism）を採用し、高い計算効率と効率的なメモリ利用を目指した。モデルの学習には、8 ノード（64 GPU）または 16 ノード（128 GPU）を利用した[21]。パイプライン並列により、Transformer のブロック（層）単位でパラメータを複数の GPU に分散配置し、さらにテンソル並列化で層内のパラメータを複数の GPU に分散配置することで、13B のモデルの学習を 1 GPU あたり 40GB のメモリに載るようにしながら、分散並列学習を行った。さらに、Megatron-LM の分散最適化（distributed optimizer）を使用することで、パラメータの最適化のために保持すべき情報をデータ並列間で分散して保持し、最適化に必要なメモリ消費量を削減した。8 ノードの実験環境では、データ並列数を 8 (DP=8)、テンソル並列数を 2 (TP=2)、パイプライン並列数を 4 (PP=4) とし、さらに系列並列化（sequence parallelism）も用いた。16 ノードの実験環境では、8 ノードの実験設定からデータ並列数を 2 倍の 16 (DP=16) に増やした。

D.3　実験設定

今回の実験は Llama 2 13B の継続事前学習であるので、言語モデルのアーキテクチャは Llama 2 13B をそのまま採用する。すなわち、埋め込み表現ベクトルの次元数は 5,120、フィードフォワード層の隠れ層の次元数は 13,824、アテンションヘッド数は 40、Transformer の層数は 40 である。継続事前学習を行う前に日本語の語彙をトークナイザーに追加しており、語彙数は Swallow と llm-jp-corpus v1.0.1 で 43,176、ClueWeb22 で 41,720 である[22]。学習のオプティマイザーには、AdamW [21] を採用し、ハイパーパラメータは $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 1.0 \times 10^{-8}$ と設定した。学習率のスケジューリングには、コサイン波形による減衰（cosine learning rate scheduler）を利用し、学習率は 1,000 ウォームアップ・ステップで最大値 1.0×10^{-4} に達し、最終的にはその 1/30 に減衰するように設定した。バッチサイズは 1,024 である。重み減衰（weight decay）に 0.1、勾配クリッピング（gradient clipping）に 1.0 を使用した。さらに、計算効率の向上と省メモリ化のため、Flash Attention [22] を採用した。

17） CC-MAIN-20230527223515-20230528013515-0000?.warc.gz の 0 から 9 までを用いた。

18） https://dumps.wikimedia.org/other/cirrussearch/

19） 量が多すぎるため、学習データ量を 1/20 に削減した。

20） https://www.csie.ntu.edu.tw/~cjlin/liblinear/

21） 実験に用いたノード数が揃わなかったのは、本論文で説明する実験以外にも同時に進めた実験があるためで、限られた計算資源・期間で有望そうな設定を優先してモデル構築を進めたからである。

22） Swallow と llm-jp-corpus v1.0.1 を継続事前学習に用いる場合、トークナイザーに追加する語彙は Swallow の統計情報を用いた。これに対し、本実験を進めた際の些細な経緯により、ClueWeb22 では ClueWeb22 の統計情報を用いたため語彙数に差が生じてしまっている。