トークナイザーの圧縮率を用いた有害コンテンツの判定法

梶浦 照乃 [2] * 山内 璃乃 [1] * 小柳 響子 [1] 東出 紗也夏 [1] 倉光 君郎 [1]
[1] 日本女子大学理学部 [2] 日本女子大学大学院理学研究科
{m1816023kt, m2016087ya}@ug.jwu.ac.jp kuramitsuk@fc.jwu.ac.jp

概要

本研究では，大規模言語モデルの学習に用いられる大規模コーパスの有害コンテンツ除去を目的として，トークナイザーの圧縮率をもとにテキストを有害/無害に判定する方法を提案する．まず，有害コンテンツから有害語彙モデルを構築する．次に，有害語彙モデルを用いてテキストをトークン化し，圧縮率を算出する．これにより，有害なコンテンツは高い圧縮率を示し，一般的な無害なコンテンツは比較的低い圧縮率を示すため，有害コンテンツの判定に利用できると考える．本論文では，提案手法を用いて大規模なテキストにおける有害コンテンツの判定を評価し，有害コンテンツを実用的な速度で正しく判定できることを確認した．

1 はじめに

大規模言語モデル（LLM）の驚異的な性能は，その事前学習に用いられるテキストデータの量 [1, 2] と品質 [3] によって決まる．まず，テキストデータの量を確保するために，Web 上のコンテンツをクローリングしてデータセットとすることが一般的である．このように集められた代表的な日本語データセットには，mC4[1)]，OSCAR[4]，Wikipedia 等があり，多くの LLM の事前学習に活用されている．

残る課題は品質である．品質は LLM の性能だけでなく，安全性や信頼性にも大きく影響するためである．ところが，Web から収集されたテキストは，品質管理が不十分であるため，差別や暴力を助長したり，読者を不愉快にしたり，不適切な内容や表現が含まれている．このような不適切なコンテンツが多ければ，学習された LLM が有害な出力をもたらすことが知られている [5]．我々は，LLM の安全性や信頼性に関わるリスクのあるテキストのことを有害コンテンツと呼ぶ．

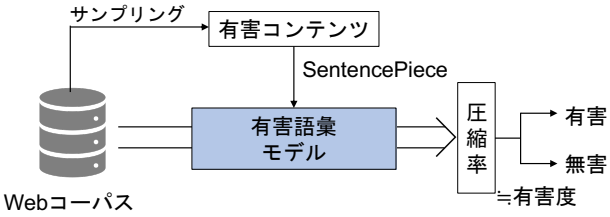

図 1 提案手法の概要

有害コンテンツへの対応法として，前処理の段階でクリーニングを処理して除去することは有効かつ必須である．しかし，日本語における有害コンテンツの判定法は，十分に確立されているとは言えない．

本研究では，LLM の学習に用いられる大規模コーパスの前処理として日本語テキストの有害度を効率よく判定する新しい手法を提案する．図 1 は我々の提案を概観している．提案では，まず大規模な Web コーパスから有害なコンテンツをサンプリングする．サンプリングした有害コンテンツをトークナイザーを用いて，有害語が多く含まれる Unigram 語彙モデルを作成する．この語彙モデルを有害語彙モデルと呼ぶ．有害語彙モデルを用いたトークナイザー[2)]を用いると，有害コンテンツはより少ないトークン長に変換されるようになる．つまり，圧縮率（１文字あたりのトークン長）で有害度が近似できる．我々は，この圧縮率から，Web コーパスの有害フィルタリングを行うことを提案する．

2 コーパスの前処理

Web コーパスは，前処理してから LLM の学習に利用することが望ましい．コーパスの前処理にはさまざまな種類があるが，有害コンテンツに利用できそうな手法は次のとおりである．

- n-gram を活用した言語識別 [6, 7]
- 繰り返し表現，記号が多すぎる行を削除 [8]

* These authors contributed equally to this work.
1） https://huggingface.co/datasets/allenai/c4
2） 有害語彙モデルは https://huggingface.co/Roy029/ngmodel_distilled_nlp2024 にて公開している．

表 1 日本語，英語，中国語，韓国語のいずれかに特化した LLM のトークナイザーを用いて Wikipedia コーパスの圧縮率を言語ごとに算出した結果

トークナイザー名	学習データ	日本語 2021	英語 2021	中国語 2018	韓国語 2021
rinna/japanese-gpt-neox-3.6b[3]	日本語 CC-100，日本語 C4，日本語 Wikipedia	0.419	0.546	-0.072	-0.370
llm-jp/hf-slow-tokenizer-v21b1[4]	日本語 Wikipedia，mC4，英語 Wikipedia，Pile，Stack	0.251	0.634	-0.208	-1.182
tokyotech-llm/Swallow-70b-instruct-hf[5]	日本語 Wikipedia，Falcon RefinedWeb，ツバメコーパス，Pile	0.201	0.646	-0.153	-0.449
meta-llama/Llama-2-7b-hf[6]	Public に利用可能なデータ (英語 89.70%)	-0.198	0.646	-0.445	-0.451
Qwen/Qwen-7B[7]	中国語，英語，多言語テキスト，コード，数学	0.244	0.711	0.233	0.280
skt/kobert-base-v1[8]	韓国語 Wikipedia，韓国語ニュース文	0.647	0.303	0.506	0.390
EleutherAI/polyglot-ko-1.3b[9]	韓国語ウェブテキスト	-0.611	0.482	-0.500	0.433

- ブロックリストによるヒューリスティックフィルタ [9, 10]
- 高品質データまたはラベル付きデータを学習させた ML ベースの品質評価手法 [7, 11]

Web コーパスはテキスト量が膨大であり，処理性能も重要な選択項目になる．その中で，正規表現を用いたブロックリストは，実装しやすく処理速度も許容範囲で，多くの LLM 開発で採用されている．また，HojiChar[10] のような実用的なツールも登場している．しかし，正規表現は有害の部分マッチによってフィルタされることが多い．例えば，「エロ」を有害語とすると，イエローも有害語として除去されてしまうなどが一例である．ほかにも，科学・医療・法律にまつわるコンテンツが無害であるにも関わらずフィルタされる傾向にあることが報告されている [12].

3 提案手法

我々は，トークナイザーの圧縮率をベースに有害コンテンツを判定する手法を新たに提案する．まずトークナイザーと語彙モデル原理について概要を述べる．続いて，本論文におけるトークナイザーの圧縮率を定義し，語彙モデルと圧縮率の相関を述べる．最後に，我々の提案手法のアイデアを述べる．

3.1 トークナイザーと語彙モデル

トークナイザーは，与えられたテキストをあらかじめ用意した語彙に基づきトークン単位に分割する．LLM のトークナイザーは，WordPiece のような定義された語彙モデルを利用するタイプと SentencePiece [13] のような学習により語彙モデルを構築するタイプに分けられる．SentencePiece では，与えられたテキストコーパスから Unigram モデルに基づいて，頻出度の高い文字列を語彙として定義し，相対的に頻度の低い語は含まれにくいような語彙モデルを構築する．

3.2 語彙モデルと圧縮率

語彙モデルを用いたトークナイザーの特徴は，テキストを分割したときの圧縮率に現れる．ここで圧縮率は，コンテンツの文字列長 L_o とトークン数 L_t の比を 1 から引いた数値である．

$$compression = 1 - \frac{L_t}{L_o}$$

圧縮率の値が大きいほど，より少ないトークン数で表現できることを示す．負の値になると，トークン化によって元の文字列長よりトークン数が増えたことを示している．また，Byte Fallback 処理が有効な場合，語彙モデルに存在しないトークンは，複数のバイトトークンに分割されて表現されるため，トークン列がさらに長くなる．

我々は，予備調査として，日本語，英語，中国語，韓国語の LLM のトークナイザーを用いて，Wikipedia コーパスに対する言語ごとの圧縮率を調査した．表 1 はその調査結果である．英語は，どの LLM でも圧縮率は高くなるが，それ以外は LLM とトークナイザーが対象としている自言語の圧縮率が高くなることが確認された．

3.3 有害語彙モデルによる判定法

我々の判定法は，トークナイザーの圧縮率に着目したものである．圧縮率は，語彙モデルに含まれるトークンが多いテキストほど高くなる傾向がある．そのため，有害語を多く含む語彙モデルを作成すれば，有害語が多いテキストは圧縮率が高くなると期待される．言い換えると，有害語彙モデルによる圧縮率は，有害率と強い相関が期待できる．

10） https://hojichar.github.io/HojiChar/hojichar.html

有害語彙モデルの利点は，ブロックリストのように有害語をひとつひとつ人手で選んで収集しなくて良い点である．有害なコンテンツをいくつかサンプリングすれば，有害語の出現頻度から語彙モデルが構築できる．新しい有害語を追加したいときも，有害コンテンツのサンプルを増やすだけで更新することが可能になる．

有害語彙モデルの特徴は，確率的な弱フィルタになることである．ブロックリストによるフィルタは，ひとつでも有害語が含まれたときフィルタすることができる強フィルタである．一方，有害語彙モデルによる圧縮率は，コンテンツ中に有害語が多く含まれるほど確率的に有害判定されるようになる．一方，有害語が少ない場合は，フィルターし損ねる可能性が高くなる弱フィルタである．

強フィルタと弱フィルタのどちらが良いかはLLM の目的によって議論すべきである．ブロックリストによる強フィルタは，どうしても有害語の誤判定が生じて，ある文字列が含まれたコンテンツが全部欠落するリスクもある．弱フィルタは，一つの語に強く反応することはなく，コンテンツ全体としての言葉使いで有害かどうか判定したいときに有用である．

4 実験

本節では，LLM の学習に用いる大規模コーパスの有害コンテンツ除去に提案手法が有効であることを確かめる．我々は，有害語彙モデルを構築し，語彙モデルの有害語の含有率と大規模コーパスに適用して有害圧縮率を確かめた．実験から，処理時間の高速性とフィルタリング性能を評価した．

4.1 有害語彙モデルの作成

我々は，mC4 日本語コーパスから HojiChar[11] が提供するブロックリストの有害語が含まれるコンテンツを 10 万件抽出した．そして，抽出したコンテンツに対して，SentencePiece トークナイザーの Unigram モデルの語彙サイズを 4k，8k, 16k, 32k と変化させて，４つの語彙モデルを作成した．

表 2 は，語彙モデルの語彙に含まれる有害語の含有率を示している．有害語は，Hojichar ブロックリストによって有害判定された語を数えたものであり，ブロックリスト外の有害語が含まれている可能性も高い．比較のため，有害コンテンツが比較的少ないと言われる Wikipedia 日本語版から構築した語彙モデルの有害語の含有率を示している．語彙モデルのサイズは，16k 以上ないと，十分に有害語が収集できないことがわかった．

表 2 有害語彙モデルの語彙の分析

トークナイザー名	語彙サイズ	有害語数	割合
megagonlabs/t5-base-japanese-web[12]	32100	221	0.688
有害語彙モデル 4k	4000	5	0.125
有害語彙モデル 8k	8000	70	0.875
有害語彙モデル 16k	16000	247	1.54
有害語彙モデル 32k	32000	590	1.84
有害濃縮語彙モデル 32k	32000	3322	10.4

我々は，もう少し含有率の高い語彙モデルを作るため，有害語のカウントを行単位で行い，最低でも 1 行に５種類以上の有害語が含まれているテキストのみ再抽出し，そこから有害濃縮語彙モデルも作成してみた．こちらは，より多くの有害語を収集できている．さらに，有害濃縮語彙モデルの語彙を目視で確認すると，HojiChar のブロックリストに含まれないが，明らかに有害と判定すべき語彙が大半であった．かなり有害語に特化した語彙モデルを作成することができた．

以後，有害濃縮語彙モデル 32k のトークナイザーを評価に用いる．

4.2 有害コンテンツの分布

我々は，有害語彙モデルのトークナイザーによる圧縮率とコンテンツの有害度を分析する．具体的には，圧縮率が高くなれば有害コンテンツであり，圧縮率が低ければ無害であることを確かめる．我々は，mC4 日本語，OSCAR23 日本語,Wikipedia 日本語から各 10 万件，ランダムにサンプリングし，それぞれの有害圧縮率を算出した．表 3 は，要約統計量をまとめたものである．図 2 は，それをヒストグラムに表示したものである．

一般に mC4 は，少なくとも日本語に関してはクリーニングされているものの有害コンテンツは多く， Wikipedia は有害コンテンツは少ないと言われ

11） https://hojichar.github.io/HojiChar/hojichar.html

5） https://huggingface.co/rinna/japanese-gpt-neox-3.6b
6） https://huggingface.co/llm-jp/llm-jp-13b-v1.0
7） https://huggingface.co/tokyotech-llm/Swallow-70b-instruct-hf
8） https://huggingface.co/meta-llama/Llama-2-7b-hf
9） https://huggingface.co/Qwen/Qwen-7B
10） https://github.com/SKTBrain/KoBERT
11） https://huggingface.co/EleutherAI/polyglot-ko-1.3b
12） https://huggingface.co/megagonlabs/t5-base-japanese-web

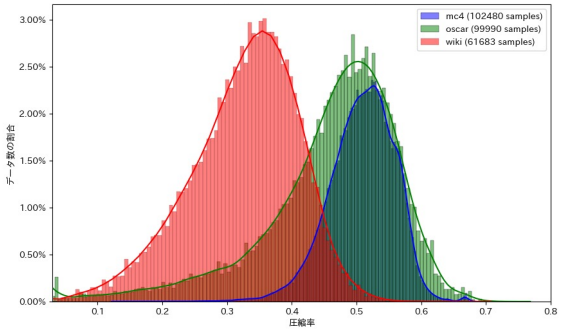

図 2　mC4 と OSCAR23，Wikipedia の圧縮率分布

表 3　日本語データセットの圧縮率の要約統計量

	第一四分位数	中央値	第三四分位数	平均値
mC4 日本語	0.460	0.510	0.550	0.508
OSCAR23 日本語	0.370	0.450	0.515	0.448
Wikipedia 日本語	0.345	0.405	0.455	0.313

る．ヒストグラムから確認できる通り，Wikipedia 日本語は圧縮率が低い傾向にあり，圧縮率がデータセットの傾向の分析に使えると言える．また，パーセンタイル（下位 n %）から選ぶことで，有害語彙の比較的少ないコンテンツだけフィルタリングすることにも適用できる．

なお，圧縮率の算出にかかる時間は，mC4 日本語 10 万件に対して 1128 秒 (約 18 分) であった．この速度は，標準的なトークナイザーの処理速度と同等で大規模なテキストをクリーニングする手法として実用的と言える．

4.3　ブロックリストとの比較

最後に，代表的なブロックリストベースのフィルターである HojiChar と提案手法を比較する．

評価データセットには，京都大学黒橋研が中心となって LLM 勉強会で作成されたベンチマークデータセット ja-mC4.valid.labeled.jsonl[13](以下，ja-mC4 ベンチマーク) を用いる．ja-mC4 ベンチマークは C4 の日本語部分 500 件に acceptable，harmful または low quality の 3 種類のラベルのアノテーションがついたものである．500 件のデータは harmful が 22 件，low quality が 245 件，acceptable が 233 件であった．我々の実験では，acceptable と low quality を無害として 2 値分類で評価する．

表 4 に，HojiChar と有害圧縮率の ja-mC4 ベンチマークでの評価結果を示す．このときの HojiChar のフィルタ設定を付録 A に示す．評価指標は，正解率，再現率，適合率，F 値とする．

13)　https://github.com/llm-jp/llm-jp-corpus

表 4　ja-mC4 ベンチマークの評価

	正解率	再現率	適合率	F 値
HojiChar	0.468	1.00	0.076	0.142
有害語彙モデル (閾値 0.520)	0.818	0.364	0.094	0.150
有害語彙モデル (閾値 0.547)	0.898	0.318	0.163	0.215
有害濃縮語彙モデル (閾値 0.474)	0.854	0.682	0.185	0.291
有害濃縮語彙モデル (閾値 0.565)	0.962	0.227	0.714	0.345

本実験では，有害圧縮率による判定のための閾値を 2 つの方法で決定して適用した．1 つ目は，ROC 曲線で左上の隅（0,1）との距離が最小となる点に設定し，閾値を 0.474 とした．2 つ目は，ROC 曲線の Youden index[14] が最大となる点に設定し，閾値を 0.565 とした．

HojiChar は強フィルタの性質があり，再現率が 1.0 と有害コンテンツを確実にブロックしている．有害圧縮率ベースの手法は，正解率と適合率，F 値において上回り，LLM の学習データ中の無害コンテンツを誤判定して削除しすぎることなく判定することができる．さらに，有害語彙モデル 32k と有害濃縮語彙モデル 32k を比較して，有害度の高いテキストをサンプリングして語彙モデルを構築することで判定性能は向上する．

5　おわりに

本研究では，LLM の学習に用いられる大規模コーパスの有害コンテンツ除去を目的として，トークナイザーの圧縮率をもとにコンテンツの有害/無害を判定する方法を提案した．我々は，SentencePiece を用いて，有害コンテンツのサンプリングから有害語が多く含まれる語彙モデルを構築することに成功した．また，有害語彙モデルを用いることで，より多くの有害語が含まれているコンテンツを優先してフィルタリングできることが確かめられた．また，処理速度は，ほとんど追加のオーバヘッドがなく，トークナイザーの処理速度でフィルタリングできることも確かめた．

今後の展望は，実際に有害語彙モデルでフィルターしたコンテンツを使って構築した LLM の性能や安全性を評価してゆきたい．

謝辞

本研究は JSPS 科研費 JP23K11374 の助成を受けたものです．2023 年度 国立情報学研究所 公募型共同研究「大規模言語モデルの効率良い学習のための訓練データ配信基盤の研究」の一部として実施されたものです．

参考文献

[1] Jared Kaplan, Sam McCandlish, T. J. Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeff Wu, and Dario Amodei. Scaling laws for neural language models. **ArXiv**, Vol. abs/2001.08361, , 2020.

[2] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. **arXiv preprint arXiv:2203.15556**, 2022.

[3] Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, et al. Textbooks are all you need. **arXiv preprint arXiv:2306.11644**, 2023.

[4] Julien Abadji, Pedro Ortiz Suarez, Laurent Romary, and Benoît Sagot. Towards a cleaner document-oriented multilingual crawled corpus. In Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Jan Odijk, and Stelios Piperidis, editors, **Proceedings of the Thirteenth Language Resources and Evaluation Conference**, pp. 4344–4355, Marseille, France, June 2022. European Language Resources Association.

[5] Julia Kreutzer, Isaac Caswell, Lisa Wang, Ahsan Wahab, Daan van Esch, Nasanbayar Ulzii-Orshikh, Allahsera Tapo, Nishant Subramani, Artem Sokolov, Claytone Sikasote, Monang Setyawan, Supheakmungkol Sarin, Sokhar Samb, Benoît Sagot, Clara Rivera, Annette Rios, Isabel Papadimitriou, Salomey Osei, Pedro Ortiz Suarez, Iroro Orife, Kelechi Ogueji, Andre Niyongabo Rubungo, Toan Q. Nguyen, Mathias Müller, André Müller, Shamsuddeen Hassan Muhammad, Nanda Muhammad, Ayanda Mnyakeni, Jamshidbek Mirzakhalov, Tapiwanashe Matangira, Colin Leong, Nze Lawson, Sneha Kudugunta, Yacine Jernite, Mathias Jenny, Orhan Firat, Bonaventure F. P. Dossou, Sakhile Dlamini, Nisansa de Silva, Sakine Çabuk Ballı, Stella Biderman, Alessia Battisti, Ahmed Baruwa, Ankur Bapna, Pallavi Baljekar, Israel Abebe Azime, Ayodele Awokoya, Duygu Ataman, Orevaoghene Ahia, Oghenefego Ahia, Sweta Agrawal, and Mofetoluwa Adeyemi. Quality at a glance: An audit of web-crawled multilingual datasets. **Transactions of the Association for Computational Linguistics**, Vol. 10, pp. 50–72, 2022.

[6] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. Fasttext. zip: Compressing text classification models. **arXiv preprint arXiv:1612.03651**, 2016.

[7] 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 Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, **Proceedings of the Twelfth Language Resources and Evaluation Conference**, pp. 4003–4012, Marseille, France, May 2020. European Language Resources Association.

[8] 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.

[9] Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. **arXiv preprint arXiv:2104.08758**, 2021.

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

[11] 吉川 克正 牧田 光晴 中町 礼文 佐藤 京也 浅原 正幸 佐藤 敏紀小林 滉河. 日本語有害表現スキーマの提案と評価. 言語処理学会第 29 回年次大会 (NLP2023), 2023.

[12] Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. 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. 1286–1305, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[13] Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In Eduardo Blanco and Wei Lu, editors, **Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations**, pp. 66–71, Brussels, Belgium, November 2018. Association for Computational Linguistics.

[14] W. J. Youden. Index for rating diagnostic tests. **Cancer**, Vol. 3, No. 1, pp. 32–35, 1950.

A　本実験における HojiChar のフィルタリング設定

Compose クラスによるテキスト処理フィルタの設定

```
from hojichar import Compose,
    document_filters

cleaner = Compose([
  document_filters.DocumentNormalizer
      (),
  # アダルト用語
  document_filters.
      DiscardAdultContentJa(),
  # 英語アダルト用語
  document_filters.
      DiscardAdultContentEn(),
  # 差別用語
  document_filters.
      DiscardDiscriminationContentJa(),
  # 暴力用語
  document_filters.
      DiscardViolenceContentJa()
])
```

LLM-jp のフィルタ実装[14]を参考に，正規表現によるパターンマッチングで除去する用語の設定

”脚注”, ”関連項目”, ”日本国内の関連項目”, ”出典”,”出典・脚注”,”参照”,”外部リンク”,”参考文献”, ”その他関連事項”
”Footnotes”, ”See also”,”Further reading”, ”Bibliography”, ”References”, ”Notes”, ”Citations”, ”Sources”,”External links”

B　評価対象のコンテンツ長と圧縮率の変化

判定対象のコンテンツ長が有害圧縮率に影響する可能性があるため，圧縮率が安定して算出できる文字数を評価する．

各 3 種類の有害/無害コンテンツを 0 から 300 文字の間で長さを変えて圧縮率の変化を観察した．有害コンテンツには，mC4 日本語のうち HojiChar で有害判定されたコンテンツを用いた．無害コンテンツには，青空文庫の小説から 3 作品を使用した．判定には，有害語彙モデル 32k を用いる．

14）https://github.com/llm-jp/llm-jp-corpus.git

結果を図 3 に示す．有害・無害に関わらずコンテンツの文字列長が 100 字を超えると，圧縮率はほぼ横ばいとなる．

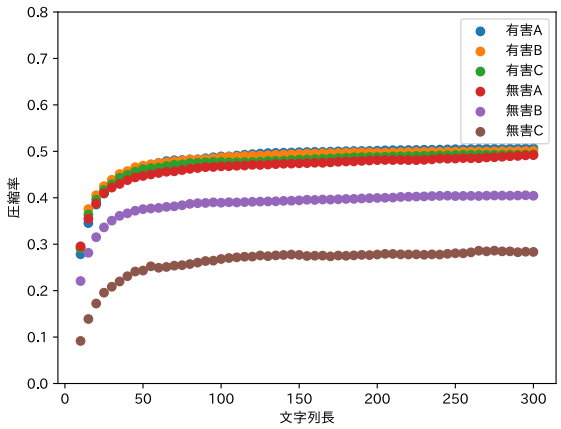

図 3　有害・無害なテキストを 0 文字から 300 文字まで変化させた場合の文字列長と圧縮率の関係．100 文字より長くなると圧縮率はほぼ横ばいになる．