日本語バイト符号化マスク言語モデルの開発と分析

工藤慧音 [1,2]　鴨田豪 [1]　塩野大輝 [1]　鈴木潤 [1,2,3]
[1] 東北大学　[2] 理化学研究所　[3] 国立情報学研究所
keito.kudo.q4@dc.tohoku.ac.jp

概要

バイト符号化を用いた日本語マスク言語モデルの開発における知見と学習したモデルの分析結果について報告する．本論文前半では，モデルアーキテクチャ・学習戦略の探索を行い，大規模言語モデルで用いられるアーキテクチャは，バイト単位マスク言語モデル構築時においても有効であること，マスク率は 50%が最適であることを確認した．後半では，学習したバイト単位マスク言語モデルの表層的な類似に対する頑健性及びモデル内部での複合化過程を分析した．学習したバイト単位マスク言語モデルは，表層的な類似に対して頑健であり，モデル内部での複合化はモデルの初期及び最終層付近で行われていることを確認した．

1　はじめに

本論文では，日本語を含むテキストをバイト列として符号化 (以下，バイト符号化とする) して学習された日本語エンコーダ型マスク言語モデル [1] (以下，バイト単位マスク言語モデルとする) に着目する．バイト符号化はテキストをバイト列として符号化する手法であり，ノイズに対する高い頑健性が注目され言語モデルの構築で用いられる [2, 3, 4]．現在，主流のサブワード単位でトークン分割 (以下，サブワード符号化とする) を行ったモデルでは，単語・フレーズごとに分割の粒度が異なるため，テキストや文字数の区切れが重要なタスクにおいて扱いにくい場合がある．一方でバイト単位マスク言語モデルは文字以下の粒度の語彙を持つため，単語分割器等の日本語基礎解析器構築，スパン予測タスク等の応用事例において有用であると期待される．

本論文の前半では，バイト単位マスク言語モデルに適したモデルアーキテクチャ・学習戦略の探索結果を報告する．特に，大規模言語モデル構築における知見がバイト単位マスク言語モデル構築においても有効であるかを検証する．また，バイト単位マスク言語モデルの学習に適したマスキング戦略を調査する．後半では，学習したバイト単位マスク言語モデルの挙動をより詳細に分析する．特に，1. 表層的には類似しているが意味の異なる単語にどの程度頑健か 2. モデル内部でどのように入力テキストが複合化 (detokenization) [5, 6] されているのか，という観点で分析する．図 1 に本研究の概要を示す．

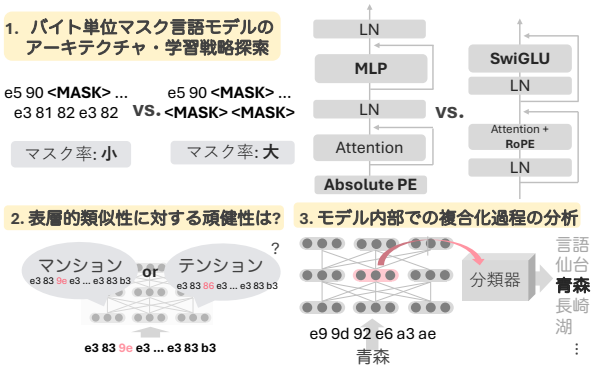

図 1　バイト単位マスク言語モデルのアーキテクチャ探索を行う，また，表層的な類似に対する頑健性と複合化の観点から，構築したモデルの挙動を分析する．

2　アーキテクチャ・学習戦略探索

バイト単位マスク言語モデルの構築において，モデルアーキテクチャや学習戦略の探索を行う．様々な設定でバイト符号化された学習データを用いたマスク言語モデリングを行いその性能を評価する．

2.1　探索空間

ベースラインモデルには，Llama2 [7] で用いられるアーキテクチャを採用する．ただし，アテンション層 (以下，Attn 層) における causal mask は適用しない．モデルの語彙数は，特殊トークンを含めて 288 としている．学習データセットとして，LLM-jp Corpus v3 [8] の一部 (117B トークン) を用いる．マスキング戦略には単語単位でまとめてマスクを適用する Whole Word Masking [9] を用いる．単語ごとに 30%の確率でマスクを適用する．その中の 80%の単

語をマスク，10%をランダムな語彙に置換，10%は変更しない．詳細は付録 A 節を参照されたい．比較対象として，以下の観点でモデルアーキテクチャや学習設定を変更し，性能を評価する．

位置符号化　絶対・相対位置符号化の 2 種類で性能を比較する．絶対位置符号化には，学習可能なパラメータで表現された位置埋め込み [1]，相対位置符号化には，Rotary Position Embedding [10] を用いる．

層正規化の位置　層正規化を残差接続後に配置する Post-LN と残差接続前に配置する Pre-LN の 2 種類が提案されており，学習の安定性では Pre-LN が有利である一方，Post-LN は学習に成功した場合の性能が高い [11]．バイト単位マスク言語モデルの学習においてどちらの配置が適しているかを比較する．

フィードフォワード層の構造　フィードフォワード層 (以下，FF 層) には，MLP [12] と SwiGLU [13] の 2 種類を用いた場合の性能を比較する．

バイアス項の有無　大規模言語モデルの構築において，学習の安定化のために各線形層のバイアス項を利用しないこともある [14]．本研究では，小規模なバイト単位マスク言語モデルの学習におけるバイアス項の有無が性能に与える影響を調査する．具体的には，1. FF 層のバイアス項を有効化 2. Attn 層のバイアス項を有効化 した場合の性能を比較する．

マスキング戦略　バイト符号化では各単語を構成するトークン数が増加するため，Whole Word Masking が有効であるかは明らかでない．そこで，1. Whole Word Masking 2. トークン単位でマスク 3. Whole Character Masking での性能を比較する．また，マスク率を変化させた時の性能も調査する．

パラメータ数とデータサイズ　データセット，パラメータ数を増加させた時の性能の変化を調査する．具体的には，パラメータ数約 100M, 200M, 400M のモデルを LLM-jp Corpus v3 からより多くの事例を用いて学習し，性能を比較する．ただし，この学習設定については計算リソースの制約から，完全に比較可能な条件での設定となっていないことに注意されたい．詳細は付録 A 節を参照のこと．

2.2　評価指標

評価データセットとして，ichikara-instruction [15]，LLM-jp Corpus v3 wikipeida ドメインの検証データセットを用いる．各事例の 15%を単語単位でランダムにマスクした文をモデルに入力し，マスクされた単語を予測できた正解率を評価指標とする．評価は

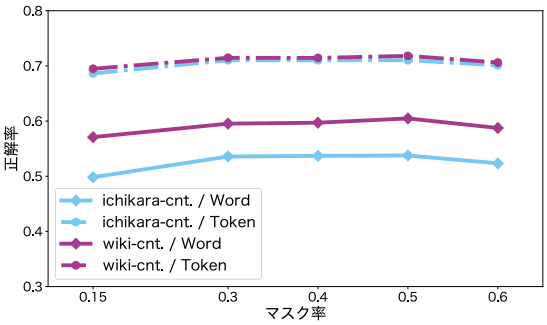

図 2　マスク率の違いと性能の関係．

cnt. 及び all の 2 種類の設定で評価を行う．cnt. ではモデル間で公平に比較するため，全ての評価対象のモデルで学習時の系列長以下となる事例のみを評価対象とする (ichikara 4606 件，wiki 594 件)．all は学習時の系列長よりも長い事例も含めて評価を行う (ichikara 4799 件, wiki 1007 件)．ただし，絶対位置符号化を用いたモデルでは，構造上学習時の系列長以上の入力は不可能であるため，各事例を学習時の系列長以下となるように分割してモデルに入力する．マスクを適用するスパンは，全ての評価対象のモデルのトークン分割器で共通して区切れとなる位置がスパンの始点，終点となるようにする．正解率は単語単位とトークン単位両方で算出する．

2.3　実験結果

表 1 に，それぞれの設定におけるマスク言語モデリングの性能を示す．比較対象として，i. 語彙数 32000 のサブワード符号化した同じデータセットで学習を行ったモデル，既存のマスク言語モデルである l. RetrievaBERT [16], m. 東北大 BERT [17] での評価も行った．学習データ量を十分の増加させることで，既存のモデルを超える性能となった (l. vs. m.)．

位置符号化手法の違い　絶対位置符号化を用いた場合に性能が低下した (a. vs. d.)．バイト単位マスク言語モデルでは，各トークンが意味に紐づかないため位置情報がより重要であると考えられる．相対位置符号化は全ての Attn 層で位置情報を埋め込む．一方で，絶対位置符号化は Transfomer の入力にのみ位置情報を埋め込むため，後半層では位置情報が比較的失われやすいと予想される．この点が絶対位置符号化での性能低下の一因となったと考えられる．

マスク手法の違い　3 つのマスク戦略 (a., b., c.) の中では a. の Whole Word Masking を用いた場合に最も高い性能となった．また，図 2 にマスク率と評価データにおける性能の関係を示す．全ての評価デー

表 1　各モデルの構造ごとの性能．Word, は単語単位，Token はトークン単位の正解率を示している．

ベースラインからの差分	Params. (M)	層数	次元数	FF 層次元数	系列長	ichikara-cnt. Word	ichikara-cnt. Token	wiki-cnt. Word	wiki-cnt. Token	ichikara-all Word	ichikara-all Token	wiki-all Word	wiki-all Token
a. ベースライン	113.7	12	768	3328	4096	53.6	71.0	59.5	71.5	53.8	71.1	34.2	45.9
b. トークン単位でマスク	113.7	12	768	3328	4096	36.5	58.3	42.0	55.6	36.6	58.3	26.0	42.8
c. Whole Character Masking	113.7	12	768	3328	4096	47.9	65.8	54.2	65.2	48.2	65.9	30.7	39.7
d. 絶対位置埋め込み	116.9	12	768	3328	4096	25.2	49.2	28.4	45.8	27.0	50.9	45.8	61.6
e. FF 層に MLP を利用	113.7	12	768	4992	4096	51.6	69.8	57.5	70.0	51.9	69.9	32.9	42.7
f. Post-LN	113.7	12	768	3328	4096	47.1	67.4	53.9	67.8	47.4	67.5	30.6	43.3
g. FF 層にバイアス項追加	113.8	12	768	3328	4096	53.4	71.0	59.8	71.6	53.7	71.1	34.8	47.5
h. Attn 層にバイアス項追加	113.7	12	768	3328	4096	53.2	70.8	59.4	71.4	53.4	70.8	43.6	56.2
i. サブワード符号化モデル	112.9	12	768	1536	4096	58.8	76.2	64.2	76.6	59.0	76.3	51.4	69.1
j. 623B tokens で学習	106.6	12	768	3072	3072	57.9	74.0	55.4	68.0	58.0	73.9	26.3	40.3
k. 637B tokens で学習	205.3	13	1024	4096	3072	60.3	75.7	57.8	70.1	60.5	75.8	33.0	45.0
l. 1.226T tokens で学習	397.0	18	1024	6144	3072	67.2	79.9	64.5	74.9	67.4	79.9	38.5	50.4
m. RetrievaBERT	1301.4	48	1536	4096	2048	62.9	61.6	62.3	60.0	63.2	61.9	62.4	60.4
n. 東北大 BERT	111.2	12	768	3072	512	54.0	55.3	64.7	63.0	53.9	55.1	59.8	58.9

タで，マスク率が 0.5 の場合に正解率が最大となった．この値は，近年のマスク言語モデル開発で用いられるマスク率 0.3-0.4 よりも高い [18, 19, 20]．

バイアス項の影響　FF 層のバイアス項を有効化した場合には性能の変化は見られなかった (a. vs. g.)．一方，Attn 層のバイアス項を有効化した場合には wiki-all において正解率が向上した (a. vs. h.)．wiki-all は平均文書長が長いことから，Attn 層のバイアス項を有効化することで，長さ対する汎化性能が向上したと考えられる．これは，大規模言語モデルの構築時の知見 [21] と一致する．

その他の比較　Post-LN に比べ Pre-LN の場合に高い性能となった (a. vs. f.)．学習が安定する Pre-LN では，学習率を大きくできたことが要因であると考えられる (詳細は付録 A)．また，FF 層には SwiGLU を採用した場合に高い性能となった (a. vs. e.)．

3　表層的な類似に対する頑健性

バイト単位マスク言語モデルでは，2 つの単語が表層的に類似している場合，入力されるトークン列の大部分は同一となる．例えば，「マンション」と「テンション」では，3 バイト目以外は同一である．モデルがこのような単語を識別することは困難である恐れがあり，下流タスクでのモデルの性能を低下させる要因となり得る．そこで，本節では入力の表層的な類似に対する頑健性を検証する．

3.1　実験設定

ある文の中に存在する単語を，表層的に類似した別の単語に置換した時の，モデルが計算する周辺文脈の生成確率の変化を調査する．単語を置換した後の周辺文脈の生成確率が大きく減少する時，モデルは 2 つの単語の意味の違いを識別できていると考えられる．表 1 の j., k., l. のモデルを分析対象とする．

評価データセット　初めに，4 つ組の単語 w_{base} (例; マンション), w_{sup} (例; テンション), w_{sem} (例; アパート), w_{unr} (例; 気分) を用意する．w_{sup} は w_{base} と表層的に類似しているが意味が異なる単語，w_{sem} は w_{base} と意味は類似しているが表層的に異なる単語，w_{unr} は w_{base} と意味も表層も異なる単語とする．また，w_{base} と w_{sem} の出現する両文脈に共通して共起すると考えられるキーワード w_{k} (例: 住む) を決める．この w_{k} を含み，w_{base} と w_{sem} が共通して解答となるような，テンプレート $q(\cdot)$ (例; 青森の{ · }に住む) を作成する．また，キーワード部分をマスクしたテンプレート $q_{\text{mask}}(\cdot)$ (例; 青森の{ · }に<mask>) も作成する．データセットの作成には Open AI o1 [22] を用いる．詳細は B 節を参照されたい．

評価指標　以下の影響度スコアを導入する．

$$I(w) = \log(P(w_{\text{k}} \mid q_{\text{mask}}(w))) - \log(P(w_{\text{k}} \mid q_{\text{mask}}(w_{\text{base}}))) \quad (1)$$

$P(\cdot)$ はモデルが計算する単語の生成確率を表す．テンプレート $q_{\text{mask}}(\cdot)$ に対して，w_{base} を当てはめた場合 (例; 青森の**マンション**に<mask>) の w_{k} の生成確率 $P(w_{\text{k}} \mid q_{\text{mask}}(w_{\text{base}}))$ を基準として，w_{sup} を当てはめた場合 (例: 青森の**テンション**に<mask>) の生成確率 $P(w_{\text{k}} \mid q_{\text{mask}}(w_{\text{sup}}))$ との差が 0 より小さいほど，w_{base} と w_{sup} を異なる意味の単語として認識できていると考えられる．また，比較対象として $I(w_{\text{sem}})$

表 2　評価データセットにおける影響度スコアの平均値.

	パラメータ数	$I(w_{\text{sup}})$	$I(w_{\text{sem}})$	$I(w_{\text{unr}})$
j.	100M	-3.29	0.49	-4.34
k.	200M	-3.92	0.52	-4.94
l.	400M	-5.18	0.25	-6.27

と $I(w_{\text{unr}})$ も算出する.

3.2　実験結果

表 2 に，データセット全体での影響度スコアの平均値を示す．実験の結果，$I(w_{\text{sup}})$ は $I(w_{\text{sem}})$ に比べて小さく，$I(w_{\text{unr}})$ に近い値となった．したがって，バイト単位マスク言語モデルは表層的な類似に対して，一定の頑健性を持っていることが示された.

4　入力テキストの複合化

バイト単位マスク言語モデルでは，入力テキストはバイト列として符号化される．そのため，モデルは推論時に入力から，文字や単語単位の情報に復元する処理を行っていると予想される．そこで，学習したモデルの隠れ状態に対してプロービングを行い，モデル内部での複合化の過程を調査する.

4.1　実験設定

学習したバイト単位マスク言語モデルの隠れ状態に対してプロービングを行い，モデル内部での複合化の過程を調べる．具体的には，モデルの特定の位置 (時刻) の隠れ状態を入力として，その位置における入力トークン (バイト) が構成要素となっている文字及び単語を予測する線形分類器を学習することができるかを検証する．表 1 の j., k., l. のモデルを分析対象とする．ベースラインとしてパラメータをランダム初期化したモデル (scratch) での評価も行う.

プロービング手法　位置 t における第 l 層の出力 (隠れ状態) を $h_{l,t}$，t 番目のトークン (バイト) を x_t とする．また，x_t が構成要素となっている文字を c_t，単語を w_t とする．例えば，入力系列が「青森」の場合，「青」,「森」はいずれも 3 バイト文字であることから，c_t = 青$(t = 0, 1, 2)$, 森$(t = 3, 4, 5)$ となる．この時，隠れ状態 $h_{l,t}$ を入力として c_t, w_t を予測する線形分類器 $f^{(l)}_{\text{char}}$ と $f^{(l)}_{\text{word}}$ を学習する．評価データセットにおける学習した分類器の文字及び単語の正解率が高い場合，その層ではバイト列から文字や単語に複合化された情報が保持されていると考えられる.

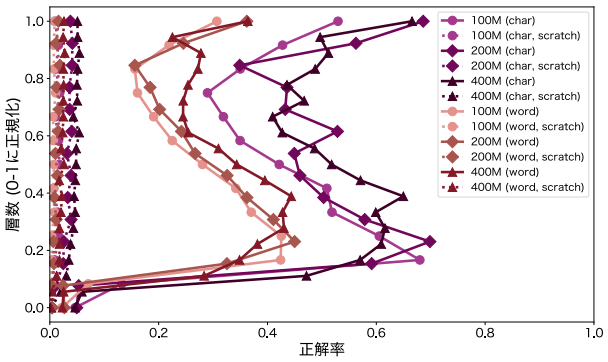

図 3　複合化の分析結果．word が単語単位，char が文字単位でのプロービングの結果である．層数 0.0 は，Transformer 層に入力される前の隠れ層 (単語埋め込み) に対してのプロービング結果である.

データセット　初めに，LLM-jp Corpus v3 を用いて出現頻度が高い 3 バイト文字及び，6 バイトで構成される単語それぞれ 500 種類からなるの語彙を作成する．文字，単語単位個別に，この語彙から一様にサンプリングして作成した長さ 3072 バイトのテキスト 10000 件を分類器の学習データ, 2000 件を評価データセットとする．サンプリングによりデータセットを構築するのは，分類器が周囲の文脈情報を加味して予測を行うことを防ぎ，複合化能力のみを純粋に評価することを可能にするためである.

4.2　実験結果

図 3 に各層の文字及び単語分類器の正解率を示す．実験の結果，初期層と最終層付近でいずれの分類器も特に高い正解率を示したことから，バイト列から文字及び単語への複合化が行われていることが示唆された.

5　おわりに

本研究では，日本語エンコーダ型マスク言語モデルの構築に向けたモデルアーキテクチャ・学習戦略を調査した．また，学習したモデルの挙動を分析し，その特性を明らかにした.

実験の結果，大規模言語モデルで用いられるアーキテクチャはバイト単位マスク言語モデルの構築においても有効であり，マスク率は 50%が最適であることが明らかとなった．さらに，学習データを増やすことで既存のモデルと同等の性能を達成した．また，学習したバイト単位マスク言語モデルは表層的な類似性に対して一定の頑健性を持っていること，モデルの初期及び最終層付近で入力テキストの複合化が行われていることが観察された.

謝辞

本研究は，JST ムーンショット型研究開発事業 JPMJMS2011-35 (fundamental research)，文部科学省の補助事業「生成 AI モデルの透明性・信頼性の確保に向けた研究開発拠点形成」,JST 国家戦略分野の若手研究者及び博士後期課程学生の育成事業（博士後期課程学生支援）JPMJBS2421 の支援を受けたものです。

本研究の一部は九州大学情報基盤研究開発センター研究用計算機システムの一般利用を利用した.

また，本研究を進めるにあたり多くの協力を賜りました Tohoku NLP グループの皆様に感謝申し上げます.

参考文献

[1] Jacob Devlin et al. 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.

[2] Linting Xue et al. ByT5: Towards a token-free future with pre-trained byte-to-byte models. **Transactions of the Association for Computational Linguistics**, Vol. 10, pp. 291–306, 2022.

[3] Junxiong Wang et al. Mambabyte: Token-free selective state space model. In **First Conference on Language Modeling**, 2024.

[4] Artidoro Pagnoni et al. Byte latent transformer: Patches scale better than tokens. **arXiv preprint arXiv:2412.09871**, 2024.

[5] Wes Gurnee et al. Finding neurons in a haystack: Case studies with sparse probing. **Transactions on Machine Learning Research**, 2023.

[6] Guy Kaplan et al. From tokens to words: On the inner lexicon of llms. **CoRR**, Vol. abs/2410.05864, , 2024.

[7] Hugo Touvron et al. Llama 2: Open foundation and fine-tuned chat models. **CoRR**, Vol. abs/2307.09288, , 2023.

[8] LLM-jp. LLM-jp Corpus v3, September 2024.

[9] Jacob Devlin et al. bert (github), 2019.

[10] Jianlin Su et al. Roformer: Enhanced transformer with rotary position embedding. **Neurocomputing**, Vol. 568, p. 127063, 2024.

[11] Sho Takase et al. B2T connection: Serving stability and performance in deep transformers. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, **Findings of the Association for Computational Linguistics: ACL 2023**, pp. 3078–3095, Toronto, Canada, July 2023. Association for Computational Linguistics.

[12] Alec Radford et al. Improving language understanding by generative pre-training. 2018.

[13] Noam Shazeer. GLU variants improve transformer. **CoRR**, Vol. abs/2002.05202, , 2020.

[14] Aakanksha et al. Palm: scaling language modeling with pathways. **J. Mach. Learn. Res.**, Vol. 24, No. 1, March 2024.

[15] 関根聡 et al. ichikara-instruction: Llm のための日本語インストラクションデータの作成. 言語処理学会 第 30 回年次大会 発表論文集, pp. 1508–1513. 言語処理学会, 3 2024.

[16] Satoru Katsumata et al. retrieva-jp/bert-1.3b, 2024.

[17] 鈴木正敏. tohoku-nlp/bert-base-japanese-char-v3, 2023.

[18] Jacob Portes et al. MosaicBERT: A bidirectional encoder optimized for fast pretraining. In **Thirty-seventh Conference on Neural Information Processing Systems**, 2023.

[19] Alexander Wettig et al. Should you mask 15% in masked language modeling? In Andreas Vlachos and Isabelle Augenstein, editors, **Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics**, pp. 2985–3000, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics.

[20] Benjamin Warner et al. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference, 2024.

[21] Jianlin Su. The magical effect of the bias term: Rope + bias = better length extrapolation, 2023.

[22] OpenAI. Openai o1 system card, 2024.

[23] 神田峻介 et al. 最小コスト法に基づく形態素解析における cpu キャッシュの効率化. 言語処理学会 第 29 回年次大会 発表論文集, pp. 345–350. 言語処理学会, 3 2023.

[24] Ilya Loshchilov et al. SGDR: stochastic gradient descent with warm restarts. In **5th International Conference on Learning Representations, ICLR, Toulon, France, April 24-26, 2017, Conference Track Proceedings**, 2017.

[25] Mohammad Shoeybi et al. Megatron-lm: Training multi-billion parameter language models using model parallelism. **CoRR**, Vol. abs/1909.08053, , 2019.

[26] 真鍋陽俊 et al. 複数粒度の分割結果に基づく日本語単語分散表現. 言語処理学会第 25 回年次大会 (NLP2019), pp. NLP2019–P8–5. 言語処理学会, 2019.

[27] В. И. Левенштейн. Двоичные коды с исправлением выпадений, вставок и замещений символов. Докл. АН СССР, Vol. 163, No. 4, pp. 845–848, 1965.

[28] Herbert E. Robbins. A stochastic approximation method. **Annals of Mathematical Statistics**, Vol. 22, pp. 400–407, 1951.

表 3 マスク言語モデル学習時のハイパーパラメータ

学習ステップ	100,000
バッチサイズ	1179648 tokens
学習率スケジューリング	Cosine [24]
学習率ウォームアップ	2000
学習率 (最大)	1×10^{-3}
学習率 (最小)	1×10^{-5}
最適化器	Adam $\beta_1 = 0.9, \beta_2 = 0.999,$ $\epsilon = 1 \times 10^{-8},$ weight_decay = 0.01
勾配クリッピング	1.0

表 4 j., k., l. 学習時のバッチサイズ (tokens) と学習ステップ数

	バッチサイズ	学習ステップ (終了値)	学習ステップ (最大値)
j.	3145728	198000	5870000
k.	2359296	270000	5870000
l.	3981312	308000	5870000

A モデル学習設定の詳細

学習データセット モデルの学習には，LLM-jp Corpus v3 [8] を用いる．表 1 の a.-i. のモデルでは，LLM-jp Corpus v3 のうち，ja_wiki, kaken, ja_warp_html のサブセットを用いた．また，j.-l. のモデルでは，LLM-jp Corpus v3 の日本語サブセット全てから，ランダムにサンプリングしたデータセットを用いた．また，Wholw Word Masking のための単語分割器として vibrato [23] を利用した．辞書には bccwj-suw+unidic-cwj-3_1_1 を用いた．

ハイパーパラメータ 表 3 にモデル学習時のハイパーパラメータを示す．ただし例外として，Post-LM モデル (表 1 f.) では学習率の最大値を 3×10^{-4}, 勾配クリッピングを 0.1 とした．これは，学習率が大きい場合に学習が安定しなかったためである．また，表 1 における j., k., l. のモデルのバッチサイズと学習ステップ数は表 4 の通りである．学習ステップ (最大値) を最大値として指定し，実際には学習ステップ (終了値) で学習を切り上げた．

実装 Megatron-LM [25] をベースとして独自の変更を加えた．

B 表層的な類似に対する頑健性評価データセット

5 節で述べたように，表層的な類似に対する頑健性を評価するためのデータセットを以下の手順で作成した．初めに，表層的に類似しているが意味が異なる単語のペア w_{base} (例; マンション) と w_{sup} (例; テンション) を用意する．具体的には，次の条件を満たす異なる単語ペアを，chive-1.1 v1.1 mc90 aunit [26] の 50000 語を語彙集合として以下の条件を満たす単語ペアを獲得する．

- レーベンシュタイン距離 [27] が 1 以下
- 2 つの単語の文字数が 5 文字以上
- cos 類似度が 0.3 以上

その後，w_{base}, w_{sem} と編集距離が 3 以上で最も cos 類似度が高い単語を w_{sem}, w_{unr} (例; アパート), (例; 気分) として選択する．

次に，w_{base} と w_{sem} が共通して当てはめることができ，かつ w_{sup} と w_{unr} は共通して当てはめることができないようなテンプレート $q(\cdot)$ (例; 青森の{ · } に住む) を作成する．この時，同時にキーワード w_{k} (例; 住む) を決める．データセットの作成には Open AI o1 [22] を用いる．4 つ組の単語とテンプレート作成に関する指示文書を入力として，テンプレートとキーワードを出力とする．ただし，選択された単語の組みによっては，自然な文でのテンプレートの作成が困難な場合もある．このような場合は，テンプレート作成が困難であることを示す出力をするように指示を与えた．作成されたデータセットの例を表 5 に示す．

表 5 表層的な類似に対する頑健性評価データセットの例．下線部がキーワード w_{k} である．

w_{base} = クラッシュ, w_{sup} = フラッシュ, w_{sem} = フリーズ, w_{unr} = flash $q(\cdot)$ = 会社のプレゼン当日に、パソコンが{ · } するトラブルが起きた。
w_{base} = バーニング, w_{sup} = モーニング, w_{sem} = シャイニング, w_{unr} = モー娘 $q(\cdot)$ = 新作 RPG では、最強の必殺技として{ · }スキルが使えます。
w_{base} = ブリスター, w_{sup} = ブラスター, w_{sem} = 外箱, w_{unr} = ファング $q(\cdot)$ = 製品を保護する梱包法方として、{ · }が一般的です。

表 6 複合化分析のための分類器の学習設定

学習データ数	10,000 事例
最適化器	SGD [28]
学習率	1.0×10^{-3} (一定)
バッチサイズ	10,000 事例
エポック数	10000

C 複合化分析のための分類器の学習設定

プロービングに用いる分類器の学習設定を表 6 に示す．