# 入力の分割単位について頑健な言語モデルの構築

清野 舜　高瀬 翔　李 聖哲　佐藤 敏紀
LINE 株式会社
{shun.kiyono, sho.takase, shengzhe.li, toshinori.sato}@linecorp.com

## 概要

本研究では，事前訓練に必要な計算資源の削減を目的として，文字とサブワード単位の両方を利用可能な言語モデルの構築に取り組む．既存のサブワード正則化技術を応用することで，文字とサブワードを同時に用いた言語モデルの事前訓練を実現する．実験では，BERT の事前訓練を題材として手法の効果を検証する．

## 1　はじめに

事前訓練済み言語モデルは NLP の各種タスクにおいて大きな成功を収めている [1]．これらのモデルにおいては，入力文をトークン列に分割する方法（以降，分割方法と言及する）がモデルに紐付いた形で規定されており，モデルのユーザは規定された分割方法に従う必要がある．

しかし，あらかじめ規定された分割方法が，ユーザが目的とするタスクの要求する分割方法と一致しない場合がある．例えば，サブワード単位で事前訓練されたモデルを，文字単位の分割を要求するタスクに適用する場合にこの問題は生じる．いま，日本語における句読点復元タスクを考える．句読点復元タスクとは，書き起こし文の読みやすさを向上させることを目的として，音声認識システムにおける後処理として用いられるタスクである [2]．ここで，句読点を挿入するべき箇所は，必ずしもサブワード分割の単位と一致しないため，文字単位での分割が必要となる．

分割方法の不一致を解決する方法として，単純には，サブワード単位と文字単位のそれぞれについて独立に事前訓練済み言語モデルを構築することが考えられる．実際，これは日本語の事前訓練済み言語モデルにおける標準的な慣習である．例えば，日本語用 BERT において，サブワード単位のもの[1)]と文字単位のもの[2)]がそれぞれ配布され，活発に用いられている．同様に，著者らも社内向けにサブワード単位・文字単位のモデルの訓練と配布を継続的におこなってきた[3)]．しかし，事前訓練は大量の計算資源を必要とする．そのため我々は，**サブワード単位と文字単位の両方を利用可能な単一の言語モデル**を訓練するための方法論を構築し，事前訓練に必要な計算資源を削減したい．

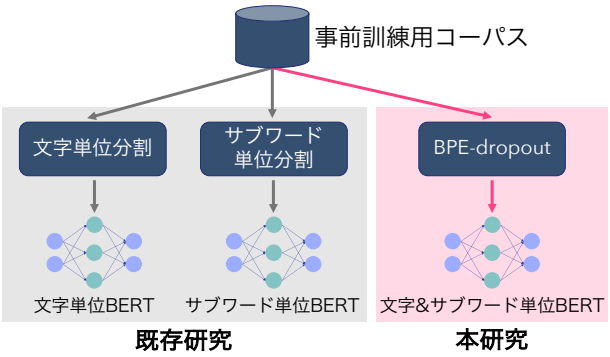


**図 1　本研究の概要図**：これまで，サブワード単位と文字単位の言語モデルが独立に作成されてきた（左）．本研究ではサブワード正則化（BPE-dropout）を応用することで，各分割単位を同時に利用可能なモデルを構築する（右）．

この目的を達成するために，本研究ではサブワード正則化 [3] を言語モデルの事前訓練に応用する（図 1）．本来，サブワード正則化は，入力文から複数の分割候補を獲得し，モデルの頑健性と汎化性能を向上させるための手法である．その代わり，本研究ではサブワード正則化をサブワード単位と文字単位を同時に言語モデルの事前訓練に取り入れるための方法論として応用する．手法そのものは非常に単純で，追加のモデルパラメータを必要としないほか，ファインチューニング時の計算コストにも全く影響しない．我々の手法の効果を示すため，BERT の事前訓練を題材として実験をおこなう．実

1）https://huggingface.co/cl-tohoku/bert-base-japanese-v2

2）https://huggingface.co/cl-tohoku/bert-base-japanese-char-v2

3）時事情報を言語モデルに反映するため，クローリングで収集したテキストデータ等を用いた言語モデルの再学習が定期的に必要となる．

験では，サブワード正則化を用いて事前訓練された BERT は，サブワード単位や文字単位について独立に訓練された BERT と同等の性能を示した．

## 2 背景

第 1 節で述べたように，我々の手法はサブワード正則化技術に基づく．本研究では，サブワード分割の手法としてバイト対符号化（Byte Pair Encoding; BPE）[4]，またサブワード正則化の手法として BPE-dropout[5] を用いる[4]．本節では，これらの手法の詳細を述べる．

### 2.1 バイト対符号化（BPE）

BPE[4] はサブワード分割手法の一種であり，入力された単語について，結合規則を繰り返し適用することで，サブワード単位の分割をおこなう．まず，入力は文字単位の系列として表現される．次に，隣接する 2 つのトークンは，結合規則表（図 2 左）に定義された規則とその優先度に応じて繰り返し結合される．例えば，図 2 において，結合規則（1）の優先度が最も高いため，この規則が最初に適用される．適用可能な結合規則を全て適用した結果が，最終的なサブワード単位の分割に相当する．

BPE における結合規則表はコーパス中の頻度情報を用いて構築される．具体的には，隣接するトークンのうち，最も頻度の大きいものが新しい結合規則として追加されていく．例えば，図 2 左の場合，表の先頭に登場する結合規則ほど優先度が高い．この過程は，結合規則の総数があらかじめ指定した数に達するまで繰り返される．

### 2.2 BPE-dropout

BPE-dropout[5] は BPE 用のサブワード正則化手法である．サブワード正則化 [3] とは，入力文を複数の候補に分割し，訓練に取り入れることで，モデルをノイズに対して頑健にするための技術である．通常の BPE と BPE-dropout の比較を図 2 に示す．

BPE-dropout は，結合規則の適用過程において，結合規則を確率 $p$ で棄却する．そのため，同じ入力単語であっても BPE-dropout を適用するたびに異なる分割結果が得られる．ここで，確率 $p$ が高いほど結合規則は棄却されやすい．例えば，$p = 1.0$ のとき，

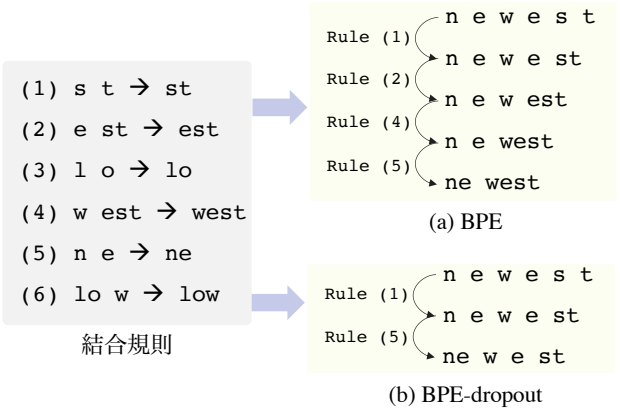


**図 2　バイト対符号化（BPE）と BPE-dropout の比較**：最初，トークン newest は文字単位の系列として表現され，結合規則表に従って結合が繰り返される．(a) BPE の場合，この処理は適用可能な結合が無くなるまで繰り返される．一方で，(b) BPE-dropout においては，結合規則が確率 $p$ で棄却される．そのため，最終的な分割結果 ne w e st は (a) BPE から得られる分割結果 ne west と異なる．

全ての結合規則が棄却されるため，分割結果は常に文字単位の分割に一致する．同様に $p = 0.0$ のとき，BPE-dropout は通常の BPE に一致する．

## 3 手法

元々，BPE-dropout のようなサブワード正則化技術は，モデルの正則化を目的として提案されたものである．一方で本研究では，同技術を応用し，サブワード単位と文字単位の両方を利用可能な言語モデルの訓練に用いる．このアイデアは，BPE-dropout による分割結果の性質に基づいている．具体的には，図 2 右において，分割結果は文字とサブワードを混合したものとなっている．このような分割を用いて訓練された言語モデルは，サブワード単位と文字単位の両方を利用可能であることが期待される．このとき，サブワードと文字単位について独立に言語モデルを訓練する必要が無くなるため，事前訓練に必要な計算資源の削減が達成できる．

我々の手法は，言語モデルの事前訓練において入力文に既存の BPE-dropout を適用するだけという非常に単純なものである．そのほか，事前訓練の目的関数やモデルのアーキテクチャの変更，モデルパラメータの追加等は一切おこなわない．事前訓練後は，所望の分割単位に合わせて BPE-dropout のパラメータ（棄却確率 $p$）を設定し，ファインチューニングをおこなう．例えば，文字単位の分割を必要とするタスクについては，$p = 1.0$ を用いる．

4) 我々の手法はサブワード正則化を活用するものであるため，同技術が利用可能であるような任意のサブワード分割手法が適用可能である．例えば，BPE の他にも，WordPiece[6, 7, 8] やユニグラム言語モデル [3] が適用できる．

## 4 実験

実験では，我々の手法の効果を日本語の BERT[1] の訓練を通して検証する．具体的には，提案手法を用いて訓練した BERT が，サブワード単位の BERT・文字単位の BERT のそれぞれとほぼ同等の性能を発揮することを示す．

### 4.1 データセット

**事前訓練用データセット** BERT の事前訓練には，日本語 Wikipedia から作成したコーパス[5] を用いた．前処理として，Unidic 辞書を用いて MeCab[6] で分かち書きをした後，BPE を用いて分割をおこなった．BPE の実装として sentencepiece[9] を用いた．語彙の大きさと文字の被覆率はそれぞれ 32,000 と 0.9995 とした．

**JGLUE データセット** 事前訓練した BERT の性能評価を目的として，JGLUE データセット [10] 上でファインチューニングを行う．JGLUE は英語圏で広く用いられているベンチマークデータである GLUE[11] の日本語版である．JGLUE のうち，JNLI, MARC-ja と JSTS の性能を報告する．JGLUE は評価セットを公開していないため，公開されている開発セットを 2 つに等分割したものを開発セットと評価セットとして用いた．

**句読点復元データセット** 文字単位での分割が必要となるタスク上での評価を目的として，句読点復元タスクにも取り組む．句読点復元タスクは，入力文に対して句読点を付与するもので，音声認識システムの後処理として用いられる [2]．本研究では，日本語の生コーパスを用いてデータセットの自動構築をおこなった．まず，CC-100 コーパス [12, 13] の日本語部分からランダムに 10 万文をサンプルし，句読点を除去した．次に，文中の各文字について，句点挿入，読点挿入とそれ以外の 3 種類のラベルを付与した．このタスクを系列ラベリング問題として定式化し，先行研究 [1] と同様に BERT のファインチューニングをおこなった．

### 4.2 比較手法

実験には，以下の 3 つの分割手法を用いて BERT の事前訓練とファインチューニングをおこなった．

- Subword: 入力文をサブワード単位で分割する手法
- Character: 入力文を文字単位で分割する手法
- BPE-dropout: 入力文を BPE-dropout を用いて分割する手法

そのほか，一般に公開されている BERT をファインチューニングした結果も報告する．事前訓練に用いるデータセット，実装やアーキテクチャの違い[7] から，我々の BERT と一般の BERT の値の大小を直接比較することはできないが，我々の BERT の性能が十分に高いことを確認する目的で用いる．

サブワード単位と文字単位の BERT として，それぞれ東北大学の公開している bert-base-japanese-v2[8] と bert-base-japanese-char-v2[9] を用いた．事前訓練中の BPE-dropout の棄却確率 $p$ は先行研究 [5] に従って 0.1 とした．その他のハイパーパラメータの詳細については付録 A を参照されたい．

### 4.3 JGLUE での結果

**BPE-dropout の効果について** 実験結果を表 1 に示した．まず，サブワード単位での性能を比較する．事前訓練に BPE-dropout を用いたモデル (c) とサブワード単位だけで事前訓練したモデル (a) の評価セット上の性能を比較すると，JNLI においては僅かに性能の劣化が見られたものの，MARC-ja と JSTS において両者はほとんど同等の性能を示した．また，文字単位での性能に着目すると，BPE-dropout を用いたモデル (d) が文字単位だけで事前訓練したモデル (b) をほぼ一貫して上回る結果となった．この結果から，BPE-dropout を BERT の事前訓練に活用することで，サブワード単位と文字単位のモデルを独立に訓練する必要がなくなるため，事前訓練に必要な計算資源を半分にできることが示唆される．

サブワード単位を用いるモデルの性能が JNLI 上で僅かに悪化した原因として，今回用いたモデル（BERT-base）が，サブワード単位と文字単位の両方について汎化するだけの表現力を持っていない可能性が考えられる．そのため，今後はより表現力を高めた大きなモデル（例：BERT-large）上での検証を

5） 2020 年 10 月に取得したダンプデータより作成した．
6） https://taku910.github.io/mecab/

7） 最も大きな違いは Layer Normalization の適用位置である．Google による BERT の元実装が Post Layer Normalization 構造を用いているのに対して，我々の実装（Megatron-LM）は Pre Layer Normazliation 構造を用いている．この差分がモデルに及ぼす影響については Takase らの研究を参照されたい [14]．
8） https://huggingface.co/cl-tohoku/bert-base-japanese-v2
9） https://huggingface.co/cl-tohoku/bert-base-japanese-char-v2

**表 1　JGLUE データセット上での性能**：JNLI と MARC-ja については正解率を報告する．また，JSTS についてはスピアマンの相関係数 $\rho$ を報告する．各値はそれぞれ 3 つの乱数シードの平均値である．

| | | | JNLI | | MARC-ja | | JSTS | |
|---|---|---|---|---|---|---|---|---|
| Model ID | Pretraining | Finetuning | Valid | Test | Valid | Test | Valid | Test |
| (a) | Subword | Subword | 88.55 | 89.43 | 95.74 | 95.19 | 85.09 | 87.71 |
| (b) | Character | Character | 85.54 | 86.91 | 94.65 | 95.08 | 82.97 | 84.75 |
| (c) | BPE-dropout | Subword | 88.00 | 88.69 | 95.54 | 95.26 | 84.52 | 87.64 |
| (d) | BPE-dropout | Character | 87.37 | 88.93 | 95.21 | 95.39 | 82.91 | 86.26 |
| (e) | Subword | Character | 86.50 | 87.78 | 94.38 | 94.69 | 80.04 | 82.36 |
| (f) | bert-base-japanese-v2 | | 89.98 | 89.54 | 95.73 | 95.58 | 84.90 | 87.35 |
| (g) | bert-base-japanese-char-v2 | | 89.54 | 89.07 | 95.05 | 94.78 | 82.29 | 85.62 |

**表 2　句読点復元タスク上での性能**：マイクロ $F_1$ 値を報告する．各値はそれぞれ 3 つの乱数シードの平均値である．

| Model ID | Pretraining | Finetuning | Valid | Test |
|---|---|---|---|---|
| (b) | Character | Character | 84.64 | 84.94 |
| (d) | BPE-dropout | Character | 85.27 | 85.46 |
| (e) | Subword | Character | 83.80 | 83.94 |

おこなう予定である．

**既存の BERT との比較**　表 1 において，我々の BERT(a)・(b) と既存の BERT(f)・(g) を比較すると，全体の傾向としてほとんど同等の性能が出せていることがわかる．この結果は，我々の訓練した BERT の性能が十分に高いこと，つまり，モデル (a)・(b) が我々の手法 (c)・(d) に対する強力なベースライン手法とみなせることを示している．

### 4.4　句読点復元タスクでの結果

表 2 に句読点復元タスクの結果を示した．表 1 と同様に，BPE-dropout を用いたモデル (d) が文字単位のモデル (b) の性能を上回った．また，サブワード単位で事前学習したモデルを文字単位でファインチューニングしたモデル (e) の性能は，最も低い結果となった．このことから，文字単位のタスクで高い性能を発揮するためには，サブワード単位のモデルを文字単位モデルとしてファインチューニング時に転用するだけでは不十分であり，文字単位を含めた事前訓練が必要であることが示唆される．

## 5　分析

**BPE-dropout は事前訓練の時間を増加させるか**　第 2.2 節で述べた通り，BPE-dropout はモデルの正則化のための技術である．そのため，BPE-dropout を用いることで，事前訓練におけるモデルの収束速度が悪化し，事前訓練の時間が増加する懸念がある．BPE-dropout を用いるねらいの一つは計算資源の節約であるため，事前訓練の時間は増加しないことが望ましい．図 3 に更新回数に対する訓練損失の変化を Subword モデルと BPE-dropout モデルのそれぞれについて示した．図より，収束速度はほぼ同じであり，事前訓練に必要な時間に大きな悪影響はないと考えられる．

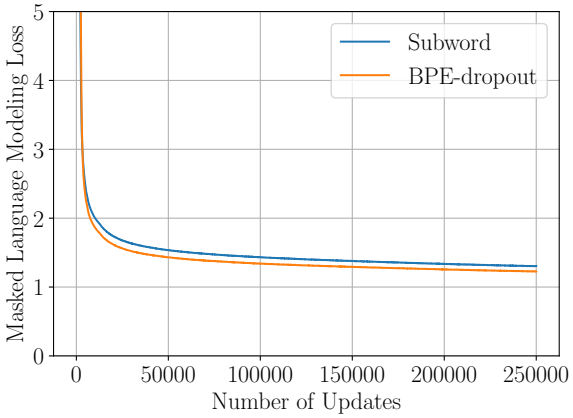


**図 3　訓練損失の変化の比較**：事前訓練時，各モデルはほぼ同じ速度で収束するとわかる．ここで，訓練データの分割方法が異なることから，損失の値の大小は直接比較できないことに注意されたい．

## 6　おわりに

本研究では，サブワード単位と文字単位の両方を利用可能な言語モデルの構築を目的として，サブワード正則化を事前訓練に取り入れる効果を検証した．BERT 上での実験結果より，サブワード正則化を既存の分割方法の代替として適用可能であることが示された．そのため，サブワードと文字単位の BERT を独立に訓練する場合と比べて，必要な計算資源を半分にできる．今後は，本手法を他の言語モデル（エンコーダ・デコーダモデル [15] やデコーダのみのモデル [16]）へ適用するための方法論の構築に取り組む予定である．

# 参考文献


[1] 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, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[2] Ottokar Tilk and Tanel Alumäe. Bidirectional recurrent neural network with attention mechanism for punctuation restoration. In **Interspeech**, pp. 3047–3051, 2016.

[3] Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. In **Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**, pp. 66–75, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[4] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In **Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**, pp. 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics.

[5] Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. BPE-dropout: Simple and effective subword regularization. In **Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics**, pp. 1882–1892, Online, July 2020. Association for Computational Linguistics.

[6] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. **arXiv preprint arXiv:1609.08144**, 2016.

[7] Xinying Song, Alex Salcianu, Yang Song, Dave Dopson, and Denny Zhou. Fast WordPiece tokenization. In **Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing**, pp. 2089–2103, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[8] Tatsuya Hiraoka. MaxMatch-dropout: Subword regularization for WordPiece. In **Proceedings of the 29th International Conference on Computational Linguistics**, pp. 4864–4872, Gyeongju, Republic of Korea, October 2022. International Committee on Computational Linguistics.

[9] Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In **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.

[10] Kentaro Kurihara, Daisuke Kawahara, and Tomohide Shibata. JGLUE: Japanese general language understanding evaluation. In **Proceedings of the Thirteenth Language Resources and Evaluation Conference**, pp. 2957–2966, Marseille, France, June 2022. European Language Resources Association.

[11] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In **Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP**, pp. 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics.

[12] 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 **Proceedings of the Twelfth Language Resources and Evaluation Conference**, pp. 4003–4012, Marseille, France, May 2020. European Language Resources Association.

[13] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. In **Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics**, pp. 8440–8451, Online, July 2020. Association for Computational Linguistics.

[14] Sho Takase, Shun Kiyono, Sosuke Kobayashi, and Jun Suzuki. On layer normalizations and residual connections in transformers. **arXiv preprint arXiv:2206.00330**, 2022.

[15] 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.

[16] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[17] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized bert pretraining approach. **arXiv preprint arXiv:1907.11692**, 2019.

[18] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-LM: Training multi-billion parameter language models using model parallelism. **arXiv preprint arXiv:1909.08053**, 2019.

[19] Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. On the stability of fine-tuning BERT: misconceptions, explanations, and strong baselines. In **9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021**. OpenReview.net, 2021.

[20] Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q. Weinberger, and Yoav Artzi. Revisiting few-sample BERT fine-tuning. In **9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021**. OpenReview.net, 2021.

# A　ハイパーパラメータ

実験（第 4 節）で用いたハイパーパラメータの一覧を表 3 に示す．それぞれの値は先行研究 [17, 18, 19, 20] による推奨値を参考にして設定した．

**表 3**　ハイパーパラメータの一覧

| | |
|---|---|
| **事前訓練** | |
| モデル | BERT-base |
| 実装 | Megatron-LM [18] |
| 最適化アルゴリズム | Adam |
| 学習率のスケジューリング | Linear warmup and decay |
| Warmup ステップ数 | 12,500 |
| 最大学習率 | 5e-4 |
| 初期学習率 | 1e-07 |
| ドロップアウト確率 | 0.1 |
| 勾配クリッピング | 1.0 |
| Weight Decay | 0.01 |
| ミニバッチサイズ | 2,048 |
| 更新回数 | 250,000 |
| 最大系列長 | 512 |
| 語彙サイズ | 32,000 |
| BPE-dropout の確率 ($p$) | 0.1 |
| **ファインチューニング** | |
| 最適化アルゴリズム | Adam |
| 学習率のスケジューリング | Linear warmup and decay |
| Warmup ステップ数 | 合計更新回数の 5% |
| 最大学習率 | 2e-5 |
| ドロップアウト確率 | 0.1 |
| 勾配クリッピング | 1.0 |
| Weight Decay | 0.01 |
| ミニバッチサイズ | 32 |
| エポック数 | 10 |