# 金融文書を用いた追加事前学習言語モデルの構築と検証

鈴木雅弘 [1]　坂地泰紀 [1]　和泉潔 [1]　石川康 [2]
[1] 東京大学大学院 工学系研究科　[2] 日興アセットマネジメント株式会社
b2019msuzuki@socsim.org
{sakaji,izumi}@sys.t.u-tokyo.ac.jp　yasushi.ishikawa@nikkoam.com

## 概要

本研究では，汎用言語コーパスを用いて事前学習を行った BERT モデルに対し，金融コーパスを用いて追加で事前学習 (追加事前学習) を行う方が有用であるか検証を行う．追加事前学習を行ったモデルを 2 つの金融テキストを用いたタスクに適用し，追加事前学習を行うことで，汎用言語コーパスによる事前学習モデルを上回る性能を持つことを示す．また，Tokenizer を構築する際のコーパスの比較を行った．追加事前学習モデルの Tokenizer に用いるコーパスにおいて，汎用言語コーパスのみによるモデルと金融コーパスを用いたモデルの間に性能差は見られなかった．

## 1　はじめに

近年，決算短信や有価証券報告書，ニュース記事や証券レポートなど，インターネットで閲覧可能な金融文書が豊富に存在する．金融関連のテキストの分析は投資やマーケット分析に役立つ一方で，毎日大量に作成されるテキストを人手によって全て分析することは難しい．そのため，近年盛んにおこなわれているのが，金融文書に自然言語処理 (NLP) を適用する金融テキストマイニングである．機械学習を用いた金融関連のツイートのセンチメント分析 [1, 2] をはじめとして，金融分野における自然言語処理に，機械学習を適用する研究が多く存在する [3, 4].

本研究では，日本語金融コーパスによって追加事前学習を行った BERT モデルについて，汎用言語コーパスから事前学習を行ったモデルと比較を行う．BERT[5] は事前学習によって各言語タスクの精度を大幅に改善したモデルである．BERT では，まず大規模言語コーパスから事前学習し，その後出力に近いレイヤーのみを学習させるファインチューニングを組み合わせる．日本語においても Wikipedia の記事から事前学習された BERT モデルが提案されている [6][1)]．しかし金融コーパスと一般的なコーパスとの間で語彙や表現の違いが大きいため，一般的なコーパスのみで学習したモデルは金融テキストマイニングのタスクに最適とは言えない．金融ドメインに適合させたモデルとして，Zuang ら [7] や鈴木ら [8] によって，金融コーパスを用いて事前学習を行ったモデルが提案されている．しかしこれらは，事前学習に適用できるコーパスが十分に大きい場合にのみ適用できる．事前学習に用いるコーパスが小さい場合にゼロから事前学習を行うと，言語モデルとして性能が低くなる可能性がある．その場合，金融ドメインのテキストデータを用いてゼロから事前学習を行うよりも，一度 Wikipedia などの汎用的な大規模コーパスを用いて事前学習を行ったモデルに対し，追加で事前学習をファインチューニングのように行う方が精度が高くなる可能性がある．そのため，本研究では Wikipedia を用いて構築された事前学習モデルに対し，金融コーパスを用いて追加で事前学習 (追加事前学習) を行う．構築したモデルと，Wikipedia のみによる事前学習モデルと比較し，追加事前学習を行うことの効果を示す．そのためにこれらのモデルに対して金融ドメインのテキストを対象とした 2 つの評価実験を行う．その際，Wikipedia から構築された Tokenizer と，Wikipedia・金融コーパスから構築された Tokenizer を用いた 2 つの追加事前学習モデルを構築し，Tokenizer を構築する際のコーパスによる性能の差の検証を行う．また金融コーパスを用いてゼロから事前学習を行ったモデルなどとも比較を行う．

本研究の貢献は以下の通りである．

- 日本語の金融コーパスで行う BERT の追加事前学習を提案する．
- 構築した追加事前学習モデルとベースラインで

1) https://github.com/cl-tohoku/bert-japanese

**表 1** 各コーパスによって構築された語彙から，「デリバティブ取引には，先物取引やスワップ取引がある」という文をトークン分割する例．"##"はサブワードに分割された語のうち，先頭でないものに付与される．

| コーパス | トークン |
|---|---|
| Wikipedia・金融 | デリバティブ / 取引 / に / は / ， / 先物 / 取引 / や / スワップ / 取引 / 等 / が / ある / ． |
| Wikipedia | デリ / ##バ / ##ティブ / 取引 / に / は / ， / 先 / ##物 / 取引 / や / スワ / ##ップ / 取引 / 等 / が / ある / ． |

ある Wikipedia による事前学習モデルの，金融テキストを用いたタスクにおける性能を比較する．

- 追加事前学習モデルにおいて，Tokenizer を構築するためのコーパスの違いがモデルの性能に与える影響を検証する．

## 2 モデルの構築

本研究では，一度 Wikipedia の日本語記事を用いて事前学習を行ったモデルに対し，金融コーパスを用いて追加事前学習を行う．BERT の事前学習は，単語の穴埋め (Masked LM) と 2 文の接続性の判定 (Next sentence prediction) の 2 つのタスクの学習によって行われる．追加事前学習でも，BERT の事前学習と同じタスクを行う．

本研究では，2 つの BERT-small モデルを構築する．1 つ目は，Wikipedia を用いて Tokenizer の構築と事前学習を行ったモデル[2]に対し，金融コーパスを用いて追加事前学習を行うモデルである．これは，汎用的な言語コーパスを用いて事前学習を行ったモデルに対し，金融ドメインを用いて追加で事前学習を行う際の標準的な学習構成と言える．2 つ目は，Wikipedia と金融コーパスを用いて Tokenizer の構築を行い，Wikipedia コーパスを用いて事前学習を行い，金融コーパスを用いて追加事前学習を行うものである．追加事前学習では，入力文をトークン列に分割する Tokenizer は事前学習と同じものを用いる必要がある．そのため 1 つ目のモデルでは Wikipedia 由来のコーパスを用いることになるが，ドメインに特化したモデルではそのドメインに適した Tokenizer を用いる方がより高い精度が得られる可能性がある．実際に構築した Tokenizer による，金融テキストのトークン分割の例を表 1 に示す．金融ドメイン特有の語である「デリバティブ」や「スワップ」について，Wikipedia・金融コーパスを用いて構築した Tokenizer では 1 トークンとして認識する．一方 Wikipedia を用いて構築した Tokenizer では，「デリ/バ/ティブ」や「スワ/ップ」のように複数トークンに分割されてしまう．1 つ目と 2 つ目のモデルの比較により，Tokenizer を構築する際に用いるコーパスによるモデルの性能を比較する．

2 つ目のモデルの構築にあたり，Wikipedia・金融コーパスを用いて構築した Tokenizer を利用して，Wikipedia を用いて事前学習を行ったモデルは公開されていない．そのため，本研究では前述の事前学習モデルの構築も行う．構築のためのパラメータには，文献 [9] で述べられている BERT-small の設定を用いる．学習率は，追加事前学習では 1e-4(土台となる BERT-small モデルの事前学習では 5e-4) とする．これは，BERT の公式リポジトリ[3]にて，追加事前学習時には事前学習より小さい学習率を用いることが推奨されているためである．

## 3 使用データ

追加事前学習に用いる金融コーパスのテキストデータとして，2 種類のデータを用いる．1 つ目は 2012 年 10 月 9 日から 2020 年 12 月 31 日にかけて開示された決算短信等のデータである．2 つ目は EDINET[4]にて，2018 年 2 月 8 日から 2020 年 12 月 31 日にかけて開示された有価証券報告書等の 2 種類データを用いる．これらのデータセットから，金融コーパス (約 2,700 万文) を作成した．

また，Wikipedia の日本語記事として，2021 年 6 月 1 日時点で作成されたダンプファイルから約 2000 万文のコーパスを作成する．

## 4 評価実験

事前学習や追加事前学習によって構築したモデルに対し，ファインチューニングによる評価実験を行い性能を評価する．本研究では 2 つの評価実験を行う．

1 つ目として，金融テキストの 1 つであるアナリストレポートを用いた実験を行う．アナリストレポートの発行日と，発行日から 12 週間後の TOPIX に対する銘柄の超過リターンの正負についての 2 値分類を行う．アナリストレポートを用いた超過リターン予測については，鈴木ら [10] が取り組んでお

2) https://huggingface.co/izumi-lab/bert-small-japanese

3) https://github.com/google-research/bert

4) https://disclosure.edinet-fsa.go.jp/

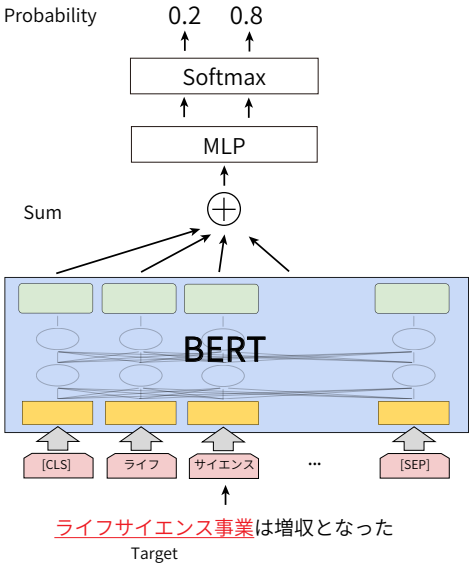


**図 1** chABSA-dataset を用いた Aspect-Based Sentiment Analysis に関する実験の概要図．入力として，「ライフサイエンス事業は増収となった」という文とセンチメントの Target「ライフサイエンス事業」が与えられる．入力文は「ライフ/サイエンス/事業/は/増収/と/なっ/た」とトークン分割され，BERT に入力される．BERT の最終層の隠れ層の出力のうち，[CLS] トークンと，Target である「ライフ」「サイエンス」「事業」の各トークンの出力が合計される．MLP と Softmax によって処理され Target のポジティブ/ネガティブの予測を行う．

り，彼らは LSTM を用いて純利益予測を行った．国内の大手証券会社が 2016 年 1 月から 2020 年 9 月にかけて発行した 78,656 本のアナリストレポートに含まれる本文を用いる．入力するアナリストレポートの本文のトークンインデックス列 $\boldsymbol{w}$ を BERT に入力する．入力トークン列 $\boldsymbol{w}$ から得られる BERT の最終層の隠れ層の出力を $\mathrm{BERT}(\boldsymbol{w}) \in \mathbb{R}^{n \times d}$ と表す．ここで $d = 256$ は BERT の隠れ層の次元数である．また BERT の出力の $i$ 番目の成分を $\mathrm{BERT}(\boldsymbol{w})_i \in \mathbb{R}^d$ と表す．BERT の出力のうち文頭 (インデックスが 0) の [CLS] トークンについての出力 $h_{\mathrm{analyst}} \in \mathbb{R}^d$ を式 (1) より得る．

$$h_{\mathrm{analyst}} = \mathrm{BERT}(\boldsymbol{w})_0 \tag{1}$$

2 値分類の出力 $y \in \mathbb{R}^2$ は式 (2)(3) によって得られる．

$$s_{\mathrm{analyst}} = \mathrm{Dropout}(\tanh(h_{\mathrm{analyst}} W_1 + b_1)) \tag{2}$$

$$y_{\mathrm{analyst}} = \mathrm{softmax}(s_{\mathrm{analyst}} W_2 + b_2) \tag{3}$$

ここで $\mathrm{Dropout}(\cdot)$ は Srivastava ら [11] が提案した Dropout を適用することを示し，$W_1 \in \mathbb{R}^{d \times d}, W_2 \in \mathbb{R}^{2 \times d}, b_1 \in \mathbb{R}^d, b_2 \in \mathbb{R}^2$ は学習可能なパラメータである．

2 つ目として，TIS 株式会社が公開している chABSA-dataset[5] を用いて，Aspect-Based Sentiment Analysis に関する実験を行う．この実験では，図 1 のように入力に文と Target 表現を入力し，その表現に関する文内でのセンチメントを出力するという問題設定にする．chABSA-dataset には，Positive, Negative, Neutral のタグが付与されていたが，Neutral が他のタグに対して大幅に少なかったことから，Neutral を除外して実験を行う．入力文をトークナイズして得られるトークン列を $\boldsymbol{w} = [w_0, w_1, \cdots, w_{n-1}]$ とする．BERT では，$w_0$ は文頭に挿入される [CLS] トークンとなる．センチメント予測の Target トークン列のインデックス番号の集合を $S \subset \{1, \cdots, n-1\}$ とする．[CLS] トークンと Target のトークン列を，式 (4) のように足し合わせ，$h_{\mathrm{chABSA}} \in \mathbb{R}^d$ を得る．

$$h_{\mathrm{chABSA}} = \sum_{i \in (\{0\} \cup S)} \mathrm{BERT}(\boldsymbol{w})_i \tag{4}$$

3 層の MLP によって式 (5)(6) のように Positive/Negative の確率 $y_{\mathrm{chABSA}} \in \mathbb{R}^2$ を出力として得る．

$$s_{\mathrm{chABSA}} = \tanh(h_{\mathrm{chABSA}} W_3 + b_3 \tag{5}$$

$$y_{\mathrm{chABSA}} = \mathrm{softmax}(s_{\mathrm{chABSA}}) W_4 + b_4) \tag{6}$$

ここで，$W_3 \in \mathbb{R}^{d \times d}, W_4 \in \mathbb{R}^{d \times 2}, b_3 \in \mathbb{R}^d, b_4 \in \mathbb{R}^2$ は学習可能なパラメータである．7,165 件のデータのうち，64%を学習データに，16%を検証データに，20%をテストデータに割り当てる．

比較として，鈴木ら [8] が構築した Wikipedia によって事前学習を行った BERT-small モデル[6] と Wikipedia・金融コーパスを用いて事前学習を行った BERT-small モデル[7] についても評価実験を行う．

## 5 結果と考察

実験結果を表 2 に示す．純利益予測では，Wikipedia と金融コーパスを組み合わせて Tokenizer を作成し，金融コーパスで追加事前学習を行ったモデルの精度が最も高かった．chABSA を用いた実験では，Wikipedia と金融コーパスを組み合わせて Tokenizer の作成と事前学習を行ったモデルが最も精度が高かった．両実験では，Wikipedia のみで事前学習を行ったモデルよりも，金融コーパスを用いた他の学習方法を採用したモデルの方が精度が高かっ

5) https://github.com/chakki-works/chABSA-dataset
6) https://huggingface.co/izumi-lab/bert-small-japanese
7) https://huggingface.co/izumi-lab/bert-small-japanese-fin

**表 2** 実験結果．F1 はマクロ平均である．学習コーパスの「Wikipedia → 金融」は，Wikipedia コーパスを用いて事前学習を行い，そこから金融コーパスを用いて追加事前学習を行うことを示す．

| 学習方法 | Tokenizer | 学習コーパス | 純利益予測 | chABSA |
|---|---|---|---|---|
| 追加事前学習 | Wikipedia | Wikipedia → 金融 | .633 | .918 |
| 追加事前学習 | Wikipedia・金融 | Wikipedia → 金融 | **.636** | .906 |
| 事前学習 [8] | Wikipedia | Wikipedia | .629 | .905 |
| 事前学習 [8] | Wikipedia・金融 | Wikipedia・金融 | .631 | **.922** |

た．これらより，金融コーパスを用いた追加事前学習の効果があったといえる．

追加事前学習モデルを行ったモデルを比較すると，純利益予測の実験では Tokenizer が Wikipedia・金融コーパスのモデルの方が精度が高かったのに対し，chABSA の実験では Tokenizer が Wikipedia のモデルの方が精度が高かった．これは，アナリストレポートを用いた純利益予測の実験では単語の認識が予測に有効だったと考えられる一方で，chABSA を用いた実験では，Target となるトークン列の範囲が入力で与えられるため，金融ドメインに特化した Tokenizer よりも，幅広い文章に対応できる Wikipedia による Tokenizer の方が精度が高くなった可能性がある．本実験においては Tokenizer の違いによる精度の差は明確ではなかった．

追加事前学習を行ったモデルと，Wikipedia・金融コーパスを用いて事前学習を行ったモデルを比較すると，純利益予測では追加事前学習を行ったモデルの方が精度が高かったのに対し，chABSA を用いたタスクでは事前学習を行ったモデルの方が精度が高かった．純利益予測ではアナリストレポートを用いているため，決算短信と有価証券報告書による金融コーパスとは文章の構成が異なっている．そのため Wikipedia を用いて一般的な日本語について事前学習を行い，その後金融コーパスにチューニングする順番によって追加事前学習モデルの精度の方が高くなった可能性がある．一方で chABSA は有価証券報告書のデータを用いているため，ゼロから有価証券報告書を含む金融コーパスで事前学習を行った方が効率よく学習できたために，事前学習モデルの精度の方が高くなったと考えられる．

アナリストレポートを用いた純利益予測ではモデル間に chABSA を用いた実験ほど差が現れなかった．これはアナリストレポートが純粋な金融テキストを対象としたタスクとしては不十分だった可能性がある．アナリストレポートを用いた純利益予測は，マーケットや情勢の急激な変化が起こった場合テキストのみから全てを正しく予測することは困難である．またアナリストによって能力の差があるため，必ずしも正しい予測ができるわけではない．更に，本実験で使用したモデルの大きさがデータセットに対して適切ではなかった可能性がある．純利益予測にて用いたアナリストレポートと，chABSA で用いたテキストデータのトークン数の中央値は，それぞれ約 500 トークンと約 55 トークンであった．BERT の small サイズでは最大 128 トークンとなるため，純利益予測では small サイズが入力可能なトークンが少なく不十分だった可能性がある．

## 6 まとめ

本研究では，汎用的言語コーパスである Wikipedia によって構築された事前学習モデルに，金融コーパスを用いた更なる事前学習である追加事前学習を提案した．Wikipedia を用いて事前学習を行ったモデルに対し，追加事前学習を行ったモデルを構築した．構築したモデルは，金融テキストを対象としたタスクにおいて Wikipedia のみから構築された事前学習モデルを上回る精度を示した．また，追加事前学習モデルにおける Tokenizer について，構築する際のコーパスの比較を行った．Wikipedia のみを用いて構築した Tokenizer によるモデルと，Wikipedia・金融コーパスを用いて構築した Tokenizer によるモデルの間に，明確な性能差は見られなかった．

今後の課題として，事前学習や追加事前学習に用いるコーパスの拡張があげられる．本研究では決算短信や有価証券報告書といった，文章のフォーマットが似ているコーパスのみを用いて追加事前学習や事前学習を行っている．金融分野に関連したニュース記事など，より幅広いコーパスを用いて学習を行うことで，より様々な表現の獲得が期待される．

## 謝辞

本研究は JSPS 科研費 JP21K12010 と JST 未来社会創造事業 JPMJMI20B1 の助成を受けたものです．

# 参考文献


[1] Nuno Oliveira, Paulo Cortez, and Nelson Areal. The impact of microblogging data for stock market prediction: Using twitter to predict returns, volatility, trading volume and survey sentiment indices. **Expert Systems with Applications**, Vol. 73, pp. 125 – 144, 2017.

[2] Gabriele Ranco, Darko Aleksovski, Guido Caldarelli, Miha Grčar, and Igor Mozetič. The effects of twitter sentiment on stock price returns. **PLoS ONE**, Vol. 10, No. 9, 2015.

[3] B. Shravan Kumar and Vadlamani Ravi. A survey of the applications of text mining in financial domain. **Knowledge-Based Systems**, Vol. 114, pp. 128–147, 2016.

[4] Li Guo, Feng Shi, and Jun Tu. Textual analysis and machine leaning: Crack unstructured data in finance and accounting. **The Journal of Finance and Data Science**, Vol. 2, No. 3, pp. 153–170, 2016.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. **CoRR**, Vol. abs/1810.04805, , 2018.

[6] 柴田知秀, 河原大輔, 黒橋禎夫. Bert による日本語構文解析の精度向上. 言語処理学会 第 25 回年次大会, 2019.

[7] Zhuang Liu, Degen Huang, Kaiyu Huang, Zhuang Li, and Jun Zhao. Finbert: A pre-trained financial language representation model for financial text mining. In Christian Bessiere, editor, **Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20**, pp. 4513–4519. International Joint Conferences on Artificial Intelligence Organization, 7 2020. Special Track on AI in FinTech.

[8] 鈴木雅弘, 坂地泰紀, 平野正徳, 和泉潔. 金融文書を用いた事前学習言語モデルの構築と検証. 人工知能学会第 27 回金融情報学研究会 (SIG-FIN), oct 2021.

[9] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. Electra: Pre-training text encoders as discriminators rather than generators, 2020.

[10] 鈴木雅弘, 堅木聖也, 坂地泰紀, 和泉潔, 石川康. テキストマイニングによるアナリストレポートを用いた株価動向予測. 言語処理学会第 26 回年次大会 (NLP2020), mar 2020.

[11] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. **CoRR**, Vol. abs/1207.0580, , 2012.