# 事前学習と追加事前学習による金融言語モデルの構築と検証

# Construction and Validation of a Pre-Training and Additional Pre-Training Financial Language Model

鈴木 雅弘[1*]　坂地 泰紀[1]　平野 正徳[1]　和泉 潔[1]
Masahiro Suzuki[1]　Hiroki Sakaji[1]　Masanori Hirano[1]　Kiyoshi Izumi[1]

[1] 東京大学大学院
[1] The University of Tokyo

**Abstract:** 本研究では決算短信や有価証券報告書を用い，言語モデルの BERT と ELECTRA について，事前学習や追加で事前学習 (追加事前学習) を行いモデルを構築する．構築したモデルについて，金融ドメインのタスクによって汎用コーパスを用いたモデルとの性能を比較する．その際，ファインチューニングを行う層の数などパラメーターによる性能についても比較を行う．構築した一部のモデルについては一般に公開する．

## 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].

近年自然言語処理の分野で用いられることの多い 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．BERT は Attention 機構をベースとした Transformer[6] によって主に構成される．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．また ELECTRA[7] は BERT に GAN[8] のアイデアを加え，GLUE において，BERT より少ない計算量で高い精度を示した．

本研究では，金融ドメインに適合した事前学習モデルと，Wikipedia を用いて構築した汎用モデルに追加で事前学習を行った追加事前学習モデルの構築を行う．構築したモデルによって金融ドメインのテキストを対象とした評価実験を行い，その結果を比較する．本研究の貢献は以下の通りである．

- 金融ドメインの文書と Wikipedia からなるコーパスを組み合わせ，事前学習や追加事前学習を行い，金融特有の専門単語を反映した金融 BERT モデルを構築した．
- 構築した各モデルを，金融ドメインで研究されているタスクを対象に実験を行い，様々な実験設定における比較を行った．

## 2　関連研究

Wikipedia や OpenWebText Corpus[1]などの汎用言語コーパスによって事前学習を行った言語モデルのパラメータは多く公開されている．英語では，Google が BERT[2]や ELECTRA[3]の事前学習モデルを公開している．日本語では，京都大学の黒橋研究室 [9] や，東北大学の乾研究室[4]が公開している BERT base モデルが存在する．これらの汎用言語コーパスによる事前学習モデルは手軽に言語モデルを構築できる一方で，特定のドメインの言語処理のタスクを行いたい場合は，公開されているパラメータを初期値とするなどして，対象ドメインのコーパスを用いて追加で事前学習を行うことが推奨されている．本研究では文献 [10] に準じ，汎用言語コーパスなどで作成した事前学習モデルに，別のコーパスでの事前学習を追加することを追加事前学習 (Additional Pre-training) と呼ぶ．Suchin らは一般的な言語コーパスで学習させた後に，対象ドメイン周

*連絡先：東京大学大学院工学系研究科
〒 113-8686 東京都文京区本郷 7-3-1
E-mail: b2019msuzuki@socsim.org

[1] https://skylion007.github.io/OpenWebTextCorpus/
[2] https://github.com/google-research/bert
[3] https://github.com/google-research/electra
[4] 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化する例．[CLS] は文頭を，[SEP] は文末などを表す．”##”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 |
|---|---|
| 金融 | [CLS] / 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 / ． / [SEP] |
| Wikipedia | [CLS] / 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 / ． / [SEP] |

辺のコーパスで重ねて事前学習を行うことで対象ドメイン内のタスクでの精度が上がることを示した [10]．この追加事前学習では，モデルのネットワークの重みは更新できるものの，入力文をトークン化する際の語彙を変更することはできない．

金融ドメインにおける事前学習モデルとして，Liu らによって提案された FinBERT[11] がある．彼らは，英語の Wikipedia や BooksCorpus に加え，FinancialWeb や YahooFinance，RedditFinance から取得したコーパスをもとに事前学習を行った．さらに，事前学習のタスクとして，オリジナルの BERT で扱う Masked LM と NSP と異なる 6 つのタスクを行っている．金融ドメインにおける追加事前学習として，仁木らによる研究 [12] や NTT データによるプレスリリース[5]がある．仁木らは Wikipedia コーパスから構築した事前学習モデルに，TIS.Inc が公開している CoARiJ[6]を用いた追加事前学習を適用した．

## 3 モデルの構築

BERT や ELECTRA といった言語モデルでは，大規模なコーパスに対してタスクを与えて事前学習 (pre-training) し，各タスクでファインチューニングするという 2 段階から構成される．

### 3.1 日本語における事前学習

BERT や ELECTRA では英語のコーパスを用いており，入力文をトークン化する際に半角スペースで分割し，その後 WordPiece[13] によるサブワード分割を行う．しかし，日本語の文章は半角スペースで分割することができない．そのため，本研究ではまず MeCab[14] によって形態素解析を行い，その後 WordPiece によるサブワード分割を行う．金融コーパスと Wikipedia のそれぞれによって構築された語彙によって，表 1 のように文をトークンに分割することが可能になる．表 1 の場合，金融コーパスによる語彙では「デリバティブ」や「先物」，「スワップ」を 1 語とし扱うのに対し，Wikipedia による語彙ではサブワードを用いて「デリ/##バ/ ##ティブ」「先/##物」「スワ/##ップ」のように分割して扱う．このように，Wikipedia による汎用的語彙には含まれないものの，金融文書においては登場する単語を，金融コーパスからモデルを作成することで扱うことができる．

サブワード分割のためのトークナイザーの学習についての実験設定は東北大学[7]によって作成されたモデルを参考にした．MeCab の辞書は IPAdic を用い，語彙数は 32,768 とした．このうち 5 語を未知語を表す [UNK]，文頭に挿入される [CLS]，2 文の間や入力の最後に挿入される [SEP]，入力長を揃えるために入力される [PAD]，Masked LM タスクの際に用いられる [MASK] に割り当てた．また，新たにファインチューニングなどの際に単語を追加するために 10 語を，1 文字の単語のために 6,129 語を割り当てた．

### 3.2 使用データ

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，3 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[8]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．3 つ目は Wikipedia の日本語記事によるコーパスである．これら 3 つのデータセットから，金融コーパス (約 4,700 万文) を作成した．金融コーパスのデータサイズは約 8GB となった．また，金融コーパスとの比較のために，Wikipedia のテキストデータ (約 2,000 万文) のみから作成したコーパスも用いる．

[5] https://www.nttdata.com/jp/ja/news/release/2020/071000/
[6] https://github.com/chakki-works/CoARiJ
[7] https://github.com/cl-tohoku/bert-japanese
[8] https://disclosure.edinet-fsa.go.jp/

表 2: 本研究で構築するモデルの一覧．学習方法の追加事前学習は，一度事前学習を行ったモデルを用い更に事前学習を行うことを示す．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．

| モデル構造 | 学習方法 | Tokenizer | 学習コーパス |
|---|---|---|---|
| BERT-base | 事前学習 | Wikipedia・金融 | Wikipedia・金融 |
| ELECTRA-base | 事前学習 | Wikipedia | Wikipedia |
| BERT-small (290 万ステップ) | 事前学習 | Wikipedia・金融 | Wikipedia・金融 |
| BERT-small | 追加事前学習 | Wikipedia | Wikipedia → 金融 |
| BERT-small | 追加事前学習 | Wikipedia・金融 | Wikipedia → 金融 |

表 3: 本研究で構築するモデルの事前学習で用いるパラメータ．Learning Rate は，10,000 ステップにおける値を示し，() 内は追加事前学習時のものを記載している．Generator Size は ELECTRA モデルにおいて，Discriminator を 1 としたときの Generator の大きさで，-は ELECTRA モデルではないため Generator の構造を持たないことを指す．

| パラメータ | BERT small | BERT base | ELECTRA base |
|---|---|---|---|
| Number of layers | 12 | 12 | 12 |
| Hidden Size | 256 | 768 | 768 |
| FFN Size | 1024 | 3072 | 3072 |
| Attention heads | 4 | 12 | 12 |
| Embedding Size | 128 | 512 | 512 |
| Learning Rate | 5e-4 (1e-4) | 1e-4 (2e-5) | 2e-4 |
| Batch Size | 128 | 256 | 256 |
| Generator Size | - | - | 1/3 |
| Train Steps | 145 万 | 100 万 | 76.6 万 |

### 3.3 実験設定

本研究で構築するモデルの一覧を表 2 に示す．最初から事前学習を行うモデルとして，Wikipedia・金融コーパスを用いた BERT-base モデルと，Wikipedia コーパスを用いた ELECTRA-base モデル，Wikipedia・金融コーパスを用い通常の 2 倍のステップ数 (290 万ステップ) の学習を行う BERT-small モデルを作成する．BERT-small モデルは，後述の追加事前学習モデルと合計の学習ステップ数を同じにするために文献 [7] の 2 倍のステップ数に渡って事前学習を行う．これらのモデルの Tokenizer は，事前学習に用いたコーパスを用いて作成する．

事前学習モデルから追加で事前学習を行う追加事前学習モデルについて，Tokenizer やコーパスの適用順を比較するため，BERT-small モデルを用いて 2 つのモデルを作成する．1 つ目は Wikipedia コーパスを用いて作成した事前学習モデルに，金融コーパスを用いて 145 万ステップ追加事前学習を行うモデルである．こちらのモデルの Tokenizer は Wikipedia コーパスから作成する．2 つ目のモデルでは，Tokenizer を Wikipedia・金融コーパスから作成し，1 つ目のモデルと同じ学習手順で，Wikipedia コーパスで事前学習し，金融コーパスで追加事前学習をする．

各モデルのパラメータは文献 [7] を参考に，表 3 のように設定した．追加事前学習では，Google の GitHub リポジトリ[9]の記載に従い，事前学習時の 5 分の 1 の学習率とし，学習ステップ数は事前学習と同じとした．Learning Rate は 10,000 ステップまで Warmup を行い，そこから線形に減衰させた．

### 3.4 実装

実装は PyTorch ベースの実装[10]を用いた．構築に用いる PyTorch ライブラリには，分散学習のためのパッケージは用意されている[11]ものの，同じ容量のメモリを持つ GPU を，ノードごとに同じ枚数利用することが前提とされている．すると，分散学習で用いる GPU のうち最も容量の小さいものにバッチサイズを合わせなくてはならず，モデル構築により多くの GPU を必要とし，計算効率が下がってしまう．そのため，本研究ではノードごとに異なる容量の GPU が利用できるようにライブラリの実装を変更する．base モデルの構築に 2 つのノードで 48GB のメモリを持つ GPU を 2 枚ずつ，2 つのノードで 32GB のメモリを持つ GPU を 2 枚ずつ用いて，それぞれの GPU メモリに合わせてバッチサイ

[9] https://github.com/google-research/bert
[10] https://github.com/huggingface/transformers
[11] https://pytorch.org/docs/stable/distributed.html

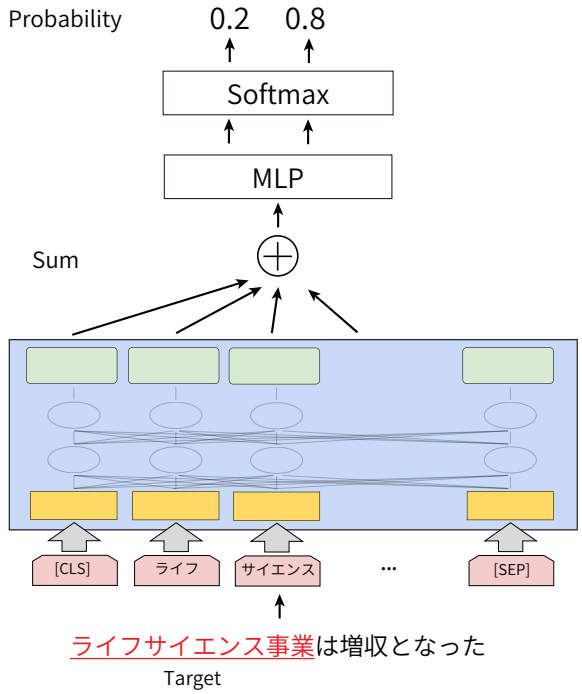


図 1: 実験の概要図．入力として,「ライフサイエンス事業は増収となった」という文とセンチメントの Target「ライフサイエンス事業」が与えられる．入力文は「ライフ/サイエンス/事業/は/増収/と/なっ/た」とトークン分割され，BERT に入力される．BERT の最終層の隠れ層の出力のうち，[CLS] トークンと，Target である「ライフ」「サイエンス」「事業」の各トークンの出力が合計される．MLP と Softmax によって処理され Target のポジティブ/ネガティブの予測を行う．

ズを設定する．なお，本研究の実験環境下では，ネットワーク速度が 10G bps であり，1G bps のネットワークに比べ学習速度は約 3.5 倍向上している．BERT-small モデルの構築には NVIDIA A6000 1 枚を用いて 2 日弱を要した．各 base モデルの構築には，32GB 以上のメモリを持つ GPU を 8 枚用いて，BERT では 14 日弱，ELECTRA モデルでは 12 日弱を要した．本実験で事前学習モデル構築のために用いたコードは，GitHub[12] にて公開している．また，構築したモデルの一部は公開している[13]．

## 4 評価実験

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，TIS 株式会社が公開している chABSA-dataset[14] 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．データセットには，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．実験では Optuna[15] によるハイパーパラメータサーチと，5-Fold の交差検証を行い，結果はそれらの平均値を用いる．また，ファインチューニングを行う層について，6 種類の方法を適用する．具体的には，分類のための線形層 (分類層とする) のみ，分類層と Transformer 層の上部 1 層，分類層と Transformer 層の上部 2 層，分類層と Transformer 層の上部 3 層，分類層と Transformer 層の上部 6 層，分類層と Transformer 層の全 (12) 層，全てのパラメータ (分類層と Transformer 層の全層，Embedding 層) のチューニングを行う．

ELECTRA モデルでは，Generator と Discriminator の 2 つのモデルが学習によって生成されるが，ELECTRA の論文 [7] に準じ，Discriminator のみを評価実験で用いる．small モデルの最大入力長は 128 とする．base モデルの最大入力長は 512 トークンであるが，学習・検証データでの最大トークン数がこれより小さかったため，256 トークンを上限とする．また，比較のために鈴木ら [15] が構築したモデルについても同様の実験を行う．

## 5 結果と考察

表 4 に実験結果を示す．small モデルの結果の平均値では，Wikipedia のみを用いて学習を行ったモデルより金融コーパスを用いて学習を行ったモデルのほうが精度が高かった．small モデルにおいては，よりドメインに特化したコーパスを用いて事前学習や追加事前学習を行うことの有効性が示された．BERT-small の構造を持ち，事前学習や追加事前学習を合計で 290 万ステップの学習を行った 3 モデル (事前学習 1 モデル，追加事前学習 2 モデル) のうち，最も精度が高かったのは追加事前学習で金融コーパスを用いたモデルとなった．このうち金融 Tokenizer を用いた 2 モデルを比較すると，事前学習モデルより追加事前学習モデルの方が精度が高かった．最初から金融コーパスとして金融テキストと Wikipedia コーパスを同時に学習させるより，初めに Wikipedia コーパスを学習させ一般的な言語表現を学習させた後に金融テキストによって金融ドメインにファインチューニングをしたことによって精度が高くなる可能性がある．また追加事前学習を行った

[12] https://github.com/retarfi/language-pretraining/tree/v1.0
[13] https://huggingface.co/izumi-lab
[14] https://github.com/chakki-works/chABSA-dataset
[15] https://github.com/optuna/optuna

表 4: 評価実験の結果．評価指標は F1 でマクロ平均を用いた．1, 2, 6, 12 はそれぞれファインチューニングを行う Transformer 層の数を表す．BERT-small2x は，BERT-small について 2 倍の (290 万) 学習ステップ数の学習を行ったことを示す．

| モデル | 学習方法 | Tokenizer | 分類層のみ | 1 | 2 | 6 | 12 | 全パラメータ | 平均 |
|---|---|---|---|---|---|---|---|---|---|
| WikiBERT-small[15] | 事前学習 | Wikipedia | .717 | .904 | .914 | .928 | .932 | .929 | .887 |
| 金融 BERT-small[15] | 事前学習 | 金融 | .809 | **.934** | .931 | .936 | .939 | .938 | .915 |
| 金融 BERT-small2x | 事前学習 | 金融 | .796 | .927 | .930 | .933 | .936 | .936 | .910 |
| 金融 BERT-small | 追加事前学習 | Wikipedia | .818 | .927 | .930 | .935 | .938 | .939 | .916 |
| 金融 BERT-small | 追加事前学習 | 金融 | **.841** | .931 | .932 | .935 | .941 | **.942** | **.920** |
| WikiELECTRA-small[15] | 事前学習 | Wikipedia | .549 | .867 | .888 | .917 | .928 | .924 | .846 |
| 金融 ELECTRA-small[15] | 事前学習 | 金融 | .714 | .921 | **.933** | **.939** | **.942** | .940 | .898 |
| WikiBERT-base[16] | 事前学習 | Wikipedia | .843 | .922 | .936 | .945 | .941 | .947 | .922 |
| 金融 BERT-base | 事前学習 | 金融 | .718 | .810 | .838 | .903 | .906 | .914 | .848 |
| 金融 BERT-base | 追加事前学習 | Wikipedia | **.887** | **.944** | **.947** | **.952** | **.952** | **.950** | **.939** |
| WikiELECTRA-base | 事前学習 | Wikipedia | .741 | .911 | .927 | .944 | .940 | .944 | .901 |
| 金融 ELECTRA-base | 事前学習 | 金融 | .625 | .802 | .850 | .897 | .881 | .893 | .825 |

BERT-small の 2 モデルを比較すると，金融 Tokenizer を用いたモデルのほうが WikipediaTokenizer を用いたモデルより精度が高かった．事前学習を行うコーパスだけでなく，Tokenizer も金融ドメインに適合させることの効果を示している．

base モデルでは追加事前学習を行った BERT モデルが最も精度が高くなった．base モデルの追加事前学習においても金融テキストを用いることで金融ドメインに適合させることの効果が示された．small モデルで Wikipedia に加えて金融テキストをコーパスとして用いることで精度が向上したのに対し，base モデルの事前学習モデルでは Wikipedia コーパスから金融コーパスに変更したことで精度が下がった．これは，base モデルのパラメータ数に対して金融テキストの量が少なかったためであると考えられる．

事前学習 BERT モデルと事前学習 ELECTRA モデルを比較すると，small･base の両モデルサイズ，Wikipedia･金融の両 Tokenizer において BERT モデルのほうが精度が高かった．これは，日本語において，BERT の事前学習タスクが，ELECTRA の事前学習タスクであるトークンの置換判定に比べて効率の良い手法であることによると考えられる．BERT で提案された Masked LM に代わるアノテーション不要な教師あり学習タスクは複数提案されている [16]．これらについて，より日本語に適した手法の採用によって精度が向上する可能性がある．

## 6 まとめ

本研究では，決算短信等のデータと有価証券報告書等のデータを Wikipedia の日本語記事と組み合わせて金融ドメインの事前学習モデルや追加事前学習モデルを構築した，また，chABSA-dataset を用いたセンチメント分析によって構築したモデルの性能を確認した．small サイズのモデルでは金融コーパスを用いて事前学習や追加事前学習を行うことで Wikipedia のみから構築した事前学習 small モデルと比較し精度が高くなった．base サイズのモデルでは金融テキストを用いて追加事前学習を行った BERT モデルは Wikipedia のみから構築した事前学習 BERT-base モデルと比べ精度が高くなった．一方で，金融コーパスを用いて事前学習を行った BERT や ELECTRA の base モデルは Wikipedia のみの事前学習モデルより精度が低くなった．

今後の課題として，base モデルで金融コーパスを用いて事前学習を行った際の精度の低下の原因を明らかにすることが考えられる．small モデルと base モデルのそれぞれで Wikipedia に加える金融テキストの量を変化させ，精度の変化を分析することで Wikipedia のみのモデルの精度を超えるために必要なテキスト量の推定が可能となると考えられる．

## 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] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30 (NeurIPS)*, pp. 5998–6008. 2017.

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

[8] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[9] 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.

[10] Suchin Gururangan, Ana Marasovic, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. *CoRR*, Vol. abs/2004.10964, , 2020.

[11] 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, 2020. Special Track on AI in FinTech.

[12] 仁木裕太, 坂地泰紀, 和泉潔, 松島裕康. 再事前学習した bert を用いた金融文書中の因果関係知識有無の判別. 人工知能学会全国大会論文集, 2020.

[13] Mike Schuster and Kaisuke Nakajima. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5149–5152, 2012.

[14] Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. Applying conditional random fields to japanese morphological analysis. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, pp. 230–237, 2004.

[15] 鈴木雅弘, 平野正徳, 坂地泰紀, 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), 2021.

[16] Atsuki Yamaguchi, George Chrysostomou, Katerina Margatina, and Nikolaos Aletras. Frustratingly simple pretraining alternatives to masked language modeling. *CoRR*, Vol. abs/2109.01819, , 2021.