Vol.2019-NL-240 No.16
2019/6/14

BERT による単語埋め込み表現列を用いた文書分類

田中 裕隆[1,a)]　曹 類[2,b)]　白 静[2,c)]　馬 ブン[2,d)]　新納 浩幸[3,e)]

概要：
BERT は双方向 Transformer の Encoder 部分を利用した事前学習モデルであり，入力文あるいは入力文対を，単語埋め込み表現列に変換する．現在，BERT を利用することで，各種の自然言語処理システムの性能が大きく向上しているが，実際のタスクに対して，BERT をどのように利用するかは個々のタスクに応じて考える必要がある．文書分類の場合，特殊 Token である [CLS] の埋め込み表現を文書の特徴ベクトルとして扱い，事前学習モデルを含めた分類器のモデル全体を Fine-Tuning する方法が標準的であるが，ここでは文書に対して BERT が出力する単語埋め込み表現列の平均ベクトルと bag of words モデルによる特徴ベクトルのそれぞれを正規化した後に，それらを連結したベクトルを作成し，それを文書の特徴ベクトルとする手法を提案する．

キーワード：BERT，文書分類，埋め込み表現，BOW

Document Classification by Word Embeddings of BERT

Tanaka Hirotaka[1,a)]　Cao Rui[2,b)]　Bai Jing[2,c)]　Ma Wen[2,d)]　Shinnou Hiroyuki[3,e)]

1. はじめに

近年，自然言語処理の多くのタスクで，事前学習モデルを利用する有効性が示されている [7][8]．事前学習モデルは様々なものが提案されているが，その中でも BERT[2] が最も優れた性能を示している．

BERT (Bidirectional Encoder Representations from Transformers) は Transformer[9] で用いられた Multi-head attention を 12 層（あるいは 24 層）重ねたモデルであり，パラメータの学習は Masked Language Model と Next Sentence Prediction という 2 つのタスクを解くことで，教師なしの枠組みの下で行われる．学習できたモデルを利用すると，入力文あるいは入力文対に対して，その単語埋め込み表現列を得ることができる．実際のタスクに対して，BERT をどのように利用するかは個々のタスクに応じて考える必要があるが，BERT の論文 [2] 中には利用法の例がいくつか示されている．例えば，SST-2 (文の感情分析) においては，入力文に対して BERT が出力する特殊 Token である [CLS] の埋め込み表現を分類器への入力として，fine tuning を行う方法が示されている．この例に従えば，文書分類については，文書全体を 1 つの文とみなして，上記の方法を利用すればよい．

ただし BERT への入力は基本的には文であり，文書に対して上記の手法が有効であるとは限らない．ここでは BERT を fine tuning ではなく，feature based な利用により文書分類を行う．具体的には文書に対して BERT が出力する単語埋め込み表現列の平均ベクトルと BOW (bag of words) モデルによる特徴ベクトルのそれぞれを正規化した後に，連結したベクトルを文書の特徴ベクトルとする手法を提案する．

[1] 茨城大学工学部情報工学科
[2] 茨城大学大学院理工学研究科情報工学専攻
[3] 茨城大学大学院理工学研究科情報科学領域
Ibaraki University, Nakanarusawa 4-12-1, Hiachi, Ibaraki 316–8511, Japan
[a)] 16t4032n@vc.ibaraki.ac.jp
[b)] 18nd305g@vc.ibaraki.ac.jp
[c)] 19nd301r@vc.ibaraki.ac.jp
[d)] 19nd302h@vc.ibaraki.ac.jp
[e)] hiroyuki.shinnou.0828@vc.ibaraki.ac.jp

実験では Webis-CLS-10[*1] の日本語の感情分析のデータを利用して，提案手法の有効性を示した．考察では BOW モデルからの特徴ベクトルと BERT からの特徴ベクトルの重みについて議論する．また BERT からの特徴ベクトルと単語の分散表現から得られる特徴ベクトルとの比較も行う．

2. 関連研究

文書分類は分類問題の一種であり，一般に教師あり学習を用いることで解決できる．そのため従来より数多くの研究がある．またディープラーニングを利用する場合でも，CNN[5] や RNN [6] を利用するなど多くの研究がある．

一方，文書分類を含め自然言語処理の多くのタスクにおいて，事前学習モデルを利用する有効性が示されている．事前学習モデルを利用する場合，大きく 2 つの利用法がある．一つは fine tuning である．これは事前学習モデルが出力する情報を，タスクを解決するためのネットワークの入力とし，その事前学習モデルを含めたネットワーク全体を学習の対象とするものである．この場合，事前学習モデルの部分は既に大量のデータから学習できた形となっているため，比較的少量のデータを用いるだけで，連結したネットワークを学習できる．OpenAI GPT [8] はニューラルネット翻訳である Transformer [9] の decoder 部分を利用した言語モデル [*2] あり，このような fine tuning の利用を念頭においている．ULMFiT [3] においても事前学習モデルを言語モデルに設定し，目的のタスクに対して fine tuning を行う．

事前学習モデルのもう一つの利用法は feature based のものである．これは事前学習モデルが出力する情報を，目的のタスクを解くための素性として利用するものである．word2vec のような単語分散表現も事前学習モデルと捉えることができる．単語の分散表現をタスク解決のための素性とした研究には文書分類を含め多くの研究がある．fastText は Subword [1] に対する分散表現を構築するが，高速かつ高精度な文書分類が行えることを実験で示している [4]．ELMo [7] は文脈を考慮した単語の分散表現を導くモデルである．実体は 2 層の双方向 LSTM であり，大規模コーパスを利用して言語モデルを学習する．これが事前学習モデルとなり，feature based の形で利用できる．

本論文で利用する BERT は従来の事前学習モデルを改善しており，様々なタスクで従来の事前学習モデルの性能を上回っている．このため本論文で扱う文書分類であっても，その効果が期待できる．ただし BERT は基本的に fine tuning の形で利用されるものであるが，文書分類では入力が文でなく文書であることから，標準的な fine tuning は有効ではないと考えられる．そのためここでは feature based で BERT を利用する．

3. 提案手法

3.1 BERT

BERT の基本のパーツは Multi-head attention である．Multi-head attention は n 単語埋め込み表現列を入力として，各埋め込み表現をより適切なものに変換して出力する．つまり出力は変換された n 単語埋め込み表現列である．

Multi-head attention の概略を述べる．基本は self attention なので Q, K, V の 3 組が入力である．今，単語埋め込み表現が m 次元であったとする．Multi-head attention では m 次元ベクトルを $d_k(= m/k)$ 次元に圧縮する線形変換器を Q, K, V それぞれに対して用意する．Q, K, V の実体は $d_k \times d_k$ の線形変換行列である．Multi-head attention の入力は n 個の m 次元ベクトルであるが，これが先の圧縮機で $n \times d_k$ の行列 X に変換され，Q, K, V に渡され $n \times d_k$ の行列 XQ, XK, XV ができる．これらを Q', K', V' とおき，以下の式 [*3] により self attention を行う．

$$\texttt{softmax}\left(\frac{Q'K'^T}{\sqrt{d_k}}\right)V'$$

これは $n \times d_k$ の行列である．上記の処理を k 個並行して行うと，$n \times d_k$ の行列が k 個作成され，これらを横に連結することで，$n \times m$ の行列が作成できる．これを更に同次元に線形変換することで Multi-head attention の出力が作られる．

BERT はこの Multi-head attention を 12 層（あるいは 24 層）重ねたモデルである．結局，BERT は n 単語埋め込み表現列を入力とし，それをより文脈に合った n 単語埋め込み表現列に変換していると捉えることができる．

3.2 [CLS] の埋め込み表現を用いた文書分類

BERT では入力となる単語列中に特殊な Token を用いる．その内の一つに [CLS] がある． [CLS] は単語列の最初に置き，これに対応する埋め込み表現を文の埋め込み表現と見なす．つまり [CLS] に対する埋め込み表現を，文の特徴ベクトルとして文の分類に使用する．

文書を BERT の入力とする場合，文書を 1 文として捉え，先頭の [CLS] の埋め込み表現を文書の特徴ベクトルとし，この特徴ベクトルを入力とした分類器を作成すれば文書分類が行える（図 1 参照）．

本論文ではこの部分の分類器（図 1 の NN に対応）には 3 層ニューラルネットワークを用いた．具体的には各層は全結合であり，順に 400 次元，50 次元，2 次元ベクトルへと線形変換される．活性化関数にはシグモイド関数を用いており，出力層に対しては softmax 関数を用いている．交差エントロピー誤差を損失関数として損失を求め，Adam によって最適化した．

[*1] https://webis.de/data/webis-cls-10.html
[*2] 言語モデルも一種の事前学習モデルである．
[*3] Scaled Dot-Product Attention

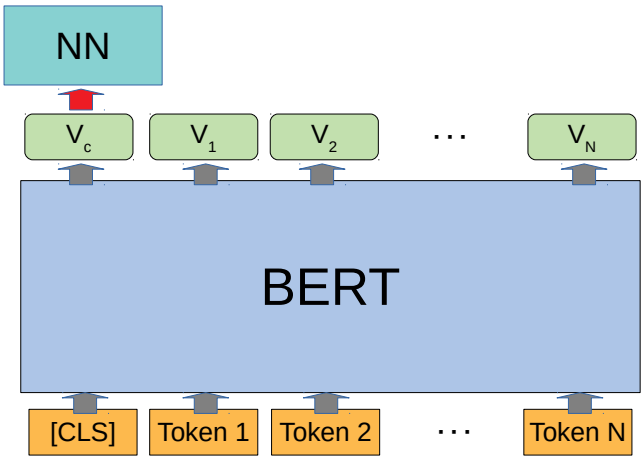

図 1 [CLS] の埋め込み表現を用いた場合

3.3 単語埋め込み表現列を用いた文書分類

BERT の入力は，基本的に 1 文（あるいは 2 文）である．入力が文書の場合でも，これを 1 つの文として扱えばよい．ただしこの場合，[CLS] の埋め込み表現が入力文書の特徴ベクトルとして適切とは限らない．

ここでは BERT の出力する単語埋め込み表現列から平均ベクトルを求め，これを文書の特徴ベクトルとして扱う（図 2 参照）．また注意として，ここで扱う単語埋め込み表現列では，BERT の特殊 Token である [CLS] と [SEP] に対する埋め込み表現を除いている．

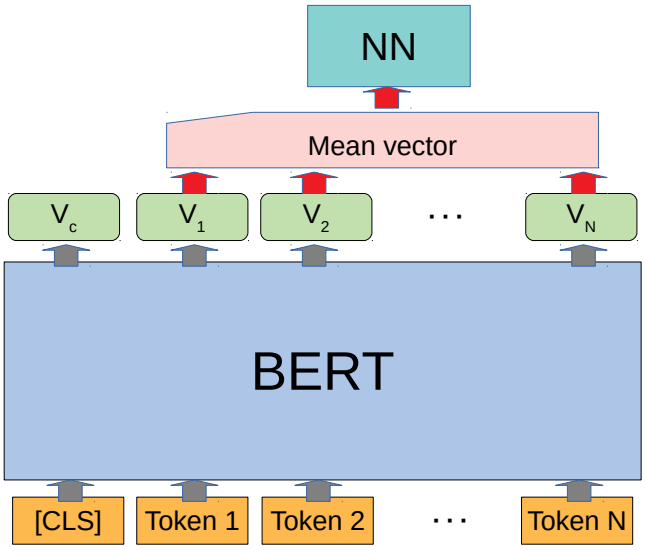

図 2 単語埋め込み表現列を用いた場合

3.4 BOW モデルとの併用

BOW (Bag-of-Words) は，ある単語表現が文書中にどの程度含まれているかを文書の特徴ベクトルとする手法である．計算手法は TF-IDF を用いる．

ここでは BERT モデルによって得られる特徴ベクトルと BOW モデルによって得られる特徴ベクトルを結合することで，文書の特徴ベクトルを構築する．ある文書 d に対して，BOW モデルによって得られる特徴ベクトルを v_b とする．次に d を形態素単位に分割し，その単語列を BERT に入力して，単語埋め込み表現列を得る．この単語埋め込み表現列から求まる平均ベクトルを v_m とする．これらの特徴ベクトル v_b と v_m は，単位ベクトルに正規化しておく．そして，v_b と v_m を連結したベクトル $[v_b; v_m]$ を d の特徴ベクトルとする（図 3 参照）．

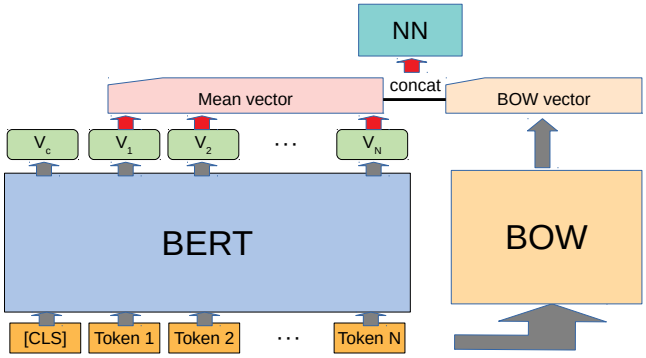

図 3 BOW モデルと併用する場合

4. 実験

4.1 実験データ

実験で使用したデータセットは，以下のサイトで公開されている Amazon のレビュー文書である．評価の 4,5 を positive，1,2 を negative とした感情分析データとして用いる．

`https://webis.de/data/webis-cls-10.html`

このデータセットは books，DVD，music の 3 つの領域がある．領域毎に訓練データに 2,000 文書，テストデータに 2,000 文書存在する．

BOW で扱う特徴語は，全訓練データ 6,000 文書に出現する 41,400 語とする．

4.2 日本語 BERT 事前モデル

公開されている BERT の多言語モデル [*4] には日本語も含まれており，日本語のタスクに対して多言語の事前学習モデルを利用することも可能である．しかし，これを利用すると基本単位が文字になってしまい，適切ではないと考えられる．そこでここでは，日本語に対応した事前学習モデルとして，京都大学黒橋・河原研究室が以下のサイトで公開している日本語事前学習モデルを使用する．

`http://nlp.ist.i.kyoto-u.ac.jp/index.php?BERT%E6%97%A5%E6%9C%AC%E8%AA%9EPretrained%E3%83%A2%E3%83%87%E3%83%AB`

また，この事前学習モデルの入力となるテキストは，同じ

[*4] `https://storage.googleapis.com/bert_models/2018_11_23/multi_cased_L-12_H-768_A-12.zip`

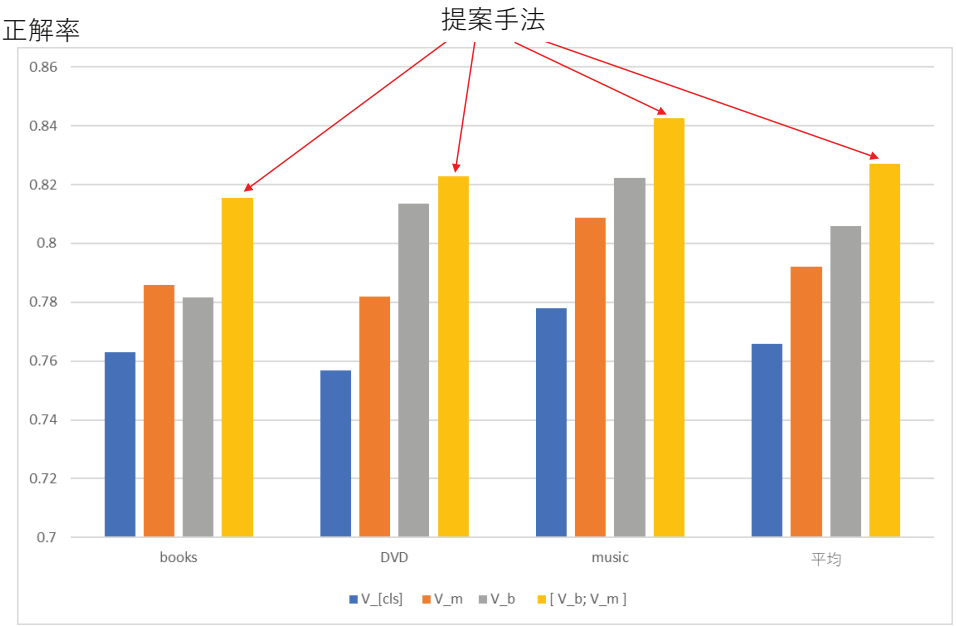

図 4 領域毎の各手法の比較

く京都大学黒橋・河原研究室が公開している Juman++[*5] で形態素解析を行い，形態素単位に分割した．

4.3 実験結果

実験結果を表 1 と図 4 に示す．$v_{[CLS]}$ は，BERT による [CLS] の埋め込み表現を用いた場合の結果である．v_m は，提案手法である BERT の単語埋め込み表現列から求めた平均ベクトルを用いた場合の結果である．[CLS] の埋め込み表現を用いるより，提案手法の方がより良い結果となった．v_b は，BOW により得られた特徴ベクトルを用いた場合の結果である．$[v_b; v_m]$ は，v_m と v_b を連結して得られた特徴ベクトルを用いる提案手法の場合の結果である．この提案手法は，実験の中で最も良い結果となった．

表 1 実験結果（各手法の正解率）

	books	DVD	music	3 領域の平均
$v_{[CLS]}$	0.7629	0.7567	0.7779	0.7658
v_m	0.7859	0.7818	0.8086	0.7921
v_b	0.7816	0.8135	0.8224	0.8058
$[v_b; v_m]$	**0.8156**	**0.8229**	**0.8427**	**0.8271**

5. 考察

5.1 BOW のベクトル と BERT のベクトルの重み

実験では，提案手法である $[v_b; v_m]$ が最も高い正解率を出した．このベクトル $[v_b; v_m]$ を作成する際に，v_b と v_m は単位ベクトルに正規化され，それぞれのベクトルの大きさの比は 1:1 としていた．

ここでは，その比を 1:2，あるいは 2:1 に変化させた場合の実験を行った．結果を表 2 に示す．v_b か v_m のどちらかに重みを加えた方が，加えないよりも良い結果が得られている．また v_m に重みを付けた方がわずかに正解率が高い．これは BERT から得られる情報の方が有用であることを示している．

表 2 v_b と v_m の重みの違い

$\|v_b\| : \|v_m\|$	books	DVD	music	3 領域の平均
1:1	0.8156	0.8229	0.8427	0.8271
1:2	**0.8160**	0.8302	**0.8455**	**0.8306**
2:1	0.8157	**0.8307**	0.8440	0.8301

5.2 分散表現列との比較

前述した実験では BERT による単語埋め込み表現列を用いた．ここでは，nwjc2vec[10] による分散表現列を用いて同様に実験を行い，その結果を比較する．nwjc2vec は，国語研日本語ウェブコーパス (NWJC) から取得した単語の分散表現データである．

入力文書に対して形態素解析を行い，得られた単語列から nwjc2vec より分散表現列を求めた．次にそれら分散表現の平均ベクトル v_e を求め正規化し，v_b と v_e を連結して $[v_b; v_e]$ を作成した．この $[v_b; v_e]$ を入力文書の特徴ベクトルとし，先の実験を行った．結果を表 3 に示す．

表 3 分散表現列を用いた手法との比較（正解率）

	books	DVD	music	3 領域の平均
v_m	0.7859	0.7818	0.8086	0.7921
$[v_b; v_m]$	**0.8156**	0.8229	**0.8427**	**0.8271**
$[v_b; v_e]$	0.7931	**0.8288**	0.8262	0.8160

[*5] http://nlp.ist.i.kyoto-u.ac.jp/index.php?JUMAN++

DVD の領域に関しては，nwjc2vec による分散表現列を用いた $[v_b; v_e]$ が高い正解率を示しているが，全体的には BERT による単語埋め込み表現列を用いた $[v_b; v_m]$ の方が良い結果である．分散表現列と BERT による単語埋め込み表現列は，同じような情報を表現しているが，BERT の方が分散表現よりも有用であることが言える．

5.3 下階層の埋め込み表現列の併用

深層学習で用いられるニューラルネットワークは，上位層より下位層の方がより一般的な概念を獲得していると考えられる．ここでは，最上位層から一つ下の層の埋め込み表現列も併用した場合の実験を行う．

BERT の最上位層の埋め込み表現から求めた平均ベクトルを v_{-1} とする．v_{-1} は，v_m と表現していたベクトルと等しい．BERT の最上位層から一つ下の層の埋め込み表現から求めた平均ベクトルを v_{-2} とする．BOW により得られた特徴ベクトルを v_b とする．これらの特徴ベクトル v_b，v_{-1}，v_{-2} は，それぞれ単位ベクトルに正規化しておく．そして特徴ベクトルを連結したベクトル $[v_b; v_{-1}; v_{-2}]$ を文書の特徴ベクトルとする．

この特徴ベクトルを用いた実験の結果を表 4 に示す．

表 4 下階層の埋め込み表現列を併用した手法との比較（正解率）

	books	DVD	music	3 領域の平均
$[v_b; v_{-1}]$	**0.8156**	0.8229	0.8427	0.8271
$[v_b; v_{-1}; v_{-2}]$	0.8141	**0.8300**	**0.8474**	**0.8305**

全体的に $[v_b; v_{-1}; v_{-2}]$ の方が $[v_b; v_{-1}]$ よりも正解率が高い．下階層の埋め込み表現列の併用することは有効である．今後はこの点を調査したい．

5.4 Fine Tuning の利用

前述したように [CLS] の埋め込み表現を用いて文書分類を行うのであれば fine tuning が行える．

ここでは BERT のソースと一緒に公開されている `run_classifier.py` [*6] を使うことで，実験データに対して fine tuning を行った．その結果を表 5 に示す．v_f が fine tuning の結果である．v_f は $v_{[CLS]}$ よりも改善されてはいるが v_m と大差ない．当然，提案手法である $[v_b; v_m]$ よりも正解率ははるかに低い．

表 5 Fine Tuning との比較（正解率）

	books	DVD	music	3 領域の平均
$v_{[CLS]}$	0.7629	0.7567	0.7779	0.7658
v_m	0.7859	0.7818	0.8086	0.7921
v_b	0.7816	0.8135	0.8224	0.8058
$[v_b; v_m]$	**0.8156**	**0.8229**	**0.8427**	**0.8271**
v_f	0.7894	0.7799	0.8019	0.7904

[*6] `https://github.com/google-research/bert`

文書分類では単純に [CLS] の埋め込み表現を用いた BERT の fine tuning よりも，提案手法のように BERT を feature based で利用する方が有効であることが確認できた．ただし本論文で行ったように，単語埋め込み表現列全体から文書の特徴ベクトルを構築し，そこから fine tuning することも可能だと思われる．今後はそれも試したい．

6. おわりに

本論文では BERT を利用して文書分類を行った．BERT を分類問題に適用する場合，特殊 Token である [CLS] の埋め込み表現を用いて BERT を含めた全体のネットワークを Fine Tuning するアプローチが自然であるが，ここでは feature based の利用を試みた．具体的には，文書に対して BERT が出力する単語埋め込み表現列の平均ベクトルと BOW モデルによる特徴ベクトルのそれぞれを正規化した後に，連結したベクトルを文書の特徴ベクトルとする．Amazon データセットを利用した実験により，提案手法の有効性を示した．今後は下階層の埋め込み表現列の併用や Fine Tuining のアプローチを試すことで，更なる精度改善を目指したい．

参考文献

[1] Bojanowski, P., Grave, E., Joulin, A. and Mikolov, T.: Enriching Word vectors with Subword Information, *Transactions of the Association for Computational Linguistics*, Vol. 5, pp. 135–146 (2017).
[2] Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, *arXiv preprint arXiv:1810.04805* (2018).
[3] Howard, J. and Ruder, S.: Universal Language Model Fine-tuning for Text Classification, *ACL-2018*, pp. 328–339 (2018).
[4] Joulin, A., Grave, E., Bojanowski, P. and Mikolov, T.: Bag of Tricks for Efficient Text Classification, *arXiv preprint arXiv:1607.01759* (2016).
[5] Kim, Y.: Convolutional Neural Networks for Sentence Classification, *EMNLP-2014*, pp. 1746–1751 (2014).
[6] Lai, S., Xu, L., Liu, K. and Zhao, J.: Recurrent convolutional neural networks for text classification, *AAAI-2015*, pp. 2267–2273 (2015).
[7] Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. and Zettlemoyer, L.: Deep Contextualized Word Representations, *NAACL-2018*, pp. 2227–2237 (2018).
[8] Radford, A., Narasimhan, K., Salimans, T. and Sutskever, I.: Improving language understanding by generative pre-training, *Technical report, OpenAI*. (2018).
[9] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. and Polosukhin, I.: Attention is all you need, *Advances in neural information processing systems*, pp. 5998–6008 (2017).
[10] 新納浩幸，浅原正幸，古宮嘉那子，佐々木稔：nwjc2vec: 国語研日本語ウェブコーパスから構築した単語の分散表現データ，自然言語処理，Vol. 24, No. 5, pp. 705–720 (2017).