四則演算を用いた Transformer の再帰的構造把握能力の調査

松本 悠太 [1]　吉川 将司 [1,2]　Benjamin Heinzerling[2]　乾 健太郎 [1,2]
[1] 東北大学　[2] 理化学研究所
yuta.matsumoto.q8@dc.tohoku.ac.jp　yoshikawa@tohoku.ac.jp
benjamin.heinzerling@riken.jp　inui@tohoku.ac.jp

概要

再帰的構造は自然言語の本質とされており，自然言語処理のモデルがこのような構造を把握する能力を持つかどうかは重要な点である．本研究では自然言語における再帰的構造の代替として，途中結果が明確な四則演算のデータを用いて Transformer の再帰的構造を把握する能力を調査する．具体的には途中の計算結果を保存することが求められるような問題に対してその値を内部表現から取り出すことができるかというタスクを通じ，Transformer が計算の途中結果を内部的に保存できていることを示す．

1　はじめに

近年，Transformer [1] の内部状態を分析し，モデルの能力を検証する研究が盛んに行われている．例えば，既存研究では品詞タグ付けや固有表現抽出によるプロービングから，モデルの下層では構文情報を，上層では意味的情報を符号化しているといったことが示されている [2, 3, 4].

さまざまな言語的特徴の中でも，再帰的な構造は自然言語において本質的な要素である [5]．再帰的構造に基づいて我々は新しい文を生成したり理解することができるためである．これは言語の構成性とも関連付けられ，既存の深層学習モデルがこの性質を捉えられるか，ベンチマークタスクを基にした調査研究が注目を集めている [6, 7].

一方，我々はプロービングの手法によって Transformer が再帰的構造を捉えることができるかについての検証を行う．その際に，自然言語の代替の形式言語として数式に着目する．これは数式もまた自然言語と同様に再帰的な構造を持ち，再帰的な構造に基づいて解かれるためである．例えば，$(a+b)\times(c+d)$ という数式を解くには一般的には $a+b, c+d$ をまず計算してから二つの計算途中結果をかけるという順番で処理をする必要があり、モデルが数式の再帰的構造を理解しているとは二つの途中結果、そして途中結果どうしを乗算した値を共に表現できているということだと定義できる．

本研究では上述のような**四則演算計算タスク**を用いて、Transformer が数式中の途中結果を保存することができるかを調査する．調査に数式を使うことには複数のメリットが存在する．数式は数値を操作するというその性質から，自然言語における句（フレーズ）などと対比して，計算の途中結果がモデル内部に保存されているかを回帰によって直接調査することができる．また，自然言語のデータセットにはバイアスが存在しており，これに起因する表面上の手がかりによってタスクを解けてしまうことが報告されている [8, 9] が，数式のような形式言語を用いることでデータセット中の分布の偏りを無くし，バイアスを抑えることができる．本タスクは学習の際に明示的に構造を教えることはしていないが構造を捉えることが本質的に要求されるタスクであり，構造の教師信号なしで構造に関する知識を符号化することができるかを測ることができる．

調査の結果，Transformer モデルは四則演算の数式を解くことができ，その際の内部表現から学習中には与えられていない計算途中の値を取り出せることがわかった．さらになぜ計算途中結果を取り出せるのかについてより詳しい調査を行ったところ，内部表現には最終的な計算結果を保存する成分とは別に計算途中結果を保存する成分が分散的に存在することが示された．

2　関連研究

自然言語処理のモデルに数値を表現させること（数量推論など）に関しては，複数の先行研究が存在する．例えば，BERT [10] のような大規模言語モデルをベースに数量推論の問題をある程度解ける [11] ということが知られている．また，自然言語処理モデルに数をテキストとして入力した際，学習データ

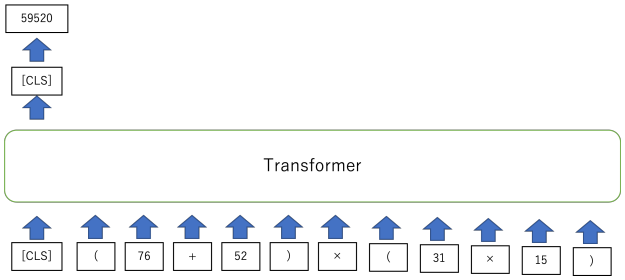


図 1: 四則演算計算タスクの概観．実際には出力は 0 から 1 の値に正規化される

に出現する数の範囲内ならばその数の大小の概念を把握することができるといったことが示されている [12, 13]．最新の数量推論に関する詳細は Thawani らのサーベイ論文 [14] を参照されたい．

自然言語における再帰的構造に起因する言語的特徴をモデルが捉えられているのかについての研究もなされてきた．Linzen らは主語と動詞の一致を通じてモデルの文法的な構造を捉える能力を検証した [15]．また，岡本らは構文情報を与えた時の LSTM [16] の内部表現の分析を行った [17]．また，深層学習モデルが構成性を捉えた汎化が可能かについての議論が盛んであるが [6, 7]，Transformer が数式を再帰的に処理可能であることを示せれば，この議論に対し肯定的な証拠を提供できるだろう．

3 問題設定

3.1 四則演算計算タスク

本研究においては図 1 のように数式をテキストとしてモデルに入力し，答えを数値として回帰させるようなタスクを考える．入力は表 1 のような 9 種類の数式で，出力は各数式の答え x_i を正規化した数である．正規化は以下の式で行われ，教師データとしては実際の答え x_i ではなく正規化された数 x_i' が使用される．

$$x_i' = \frac{x_i - \min_i x_i}{\max_i x_i - \min_i x_i} \tag{1}$$

このような計算の答えを正しく出力するためには，**(1) 数の順序構造や足し算，掛け算の概念を把握すること，(2) 計算の木構造を把握し，計算の途中結果を保存すること**が必要となる．[1)]

1) データセット中に掛け算を導入することで、演算子の優先順序ができるように工夫した．

表 1: 入力される数式のテンプレート

$a \times b$	$a \times b + c$	$a + b \times c$
$(a + b) \times c$	$a \times (c + d)$	$a + b \times (c + d)$
$(a + b) \times (c + d)$	$a + b \times c + d$	$(a + b) \times c + d$

3.2 計算途中結果保存プロービングタスク

3.1 節で示したタスクを学習する際，モデルは最終的な計算結果のみが教師データとして与えられており，その計算の途中結果は与えられていない．今回我々は $(a+b)\times(c+d)$ 中の $a+b$ の値のような，明示的に学習させていないが計算を適切に行うためには保存することが必要だと思われる計算の途中結果に着目し，このような値がモデルに保存されているかを検証する．具体的には，3.1 節のタスクで学習したモデルに対して，その隠れ層の内部表現から計算の途中結果を線形回帰させることができるかを検証する．より詳細な設定は以下各実験で詳述する．

4 実験

4.1 実験設定

データセット　訓練データ，検証データには表 1 にある 9 パターンの数式と正規化されたその答えを使用する．訓練，検証用データのサイズはそれぞれ 9 万，1 万であり，2 つのデータに登場する数値に重複はない．数式中に登場する項 a, b, c, d は全て 1,000 未満の正の実数である．また，プロービング用に $(a + b) \times (c + d)$ の形のみのデータを 3,000 件用意した．このデータ中の数式の項 a, b, c, d は全て 100 以上 1,000 未満の自然数である．これは後に内部表現の分析をするに当たって，桁数を揃えてベクトルの次元ごとの役割の対応を取るためである．

性能評価　性能評価には決定係数 R^2 を使用する．R^2 は観測値 y_i と予測値 $\hat{y}_i$ について以下の式で表される．

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \tag{2}$$

ただし，$\bar{y}$ は観測値 y_i の平均である．この値が 1 に近いほど正確な回帰ができていることを表す．

モデル　モデルには Poor Man's BERT [18] を使用する．これは BERT [10] の下位 6 層を使用したモデルで，[CLS] トークンから線形回帰を行う．また，入力は BERT を用いた数量推論で有効性が示されている Geva ら [11] の手法に従い，数字については 1

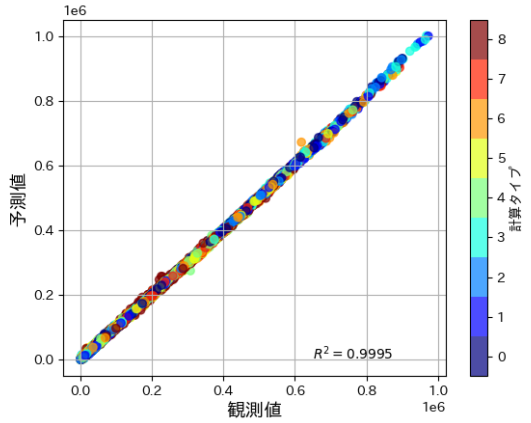


図 2: 四則演算計算タスクの回帰結果．いずれの計算タイプの数式も解くことができている．

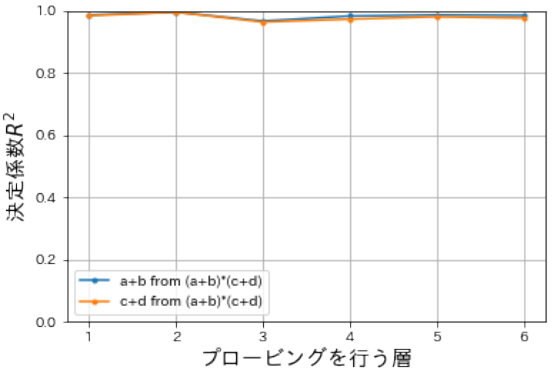


図 3: 計算途中結果保存プロービングタスクの結果．どの層の隠れ表現からでも計算の途中結果を回帰することができている．

桁ずつ分割を行う．例えば，123 という数字が入力される際，1, ##2, ##3 のように分割される．

プロービング　数式 $X = (x_1, x_2, \cdots, x_n)$ を学習済み Poor Man's BERT に入力した際の N 層目の内部表現を $H_N = (h_1, h_2, \cdots, h_n)$ とする．ここで，H_N の各ベクトル（768 次元）を連結させて新たに（$768 \times n$）次元のベクトルを作る．4.3 節で行うプロービングの際はこの結合済みベクトルから途中結果の値を線形回帰器によって回帰させる．

4.2　四則演算計算タスク

訓練データ上でモデルを学習させた際の検証データにおける回帰結果は図 2 のようになり，決定係数は 0.9995 と非常に高い値であった．図 2 から，9 つの異なるパターンを同時に学習させても，各数式の意味や大小関係を理解し，解くことができていることがわかる．次節以降では，ここで学習したモデルについてプロービングを行う．

4.3　計算途中結果保存プロービングタスク

本節では表 1 で示されている 9 つのパターンの数式のうち $(a+b)\times(c+d)$ のタイプに着目し，この数式の計算途中結果である $a+b$ や $c+d$ をモデルの各層ごとの内部表現 H_N を $n = 23$ として[2)]連結した 17,664 次元のベクトルから取り出すことができるかを検証する．できるだけ内部表現自体の能力を測り，新しく学習で教えないようにするためプロービングデータ中の 1%を訓練データ，残りの 99%を評価データとしてプロービングを行った．その結果を図 3 に示す．図 3 から，わずかなデータポイントからでも計算の途中結果である $a+b$ や $c+d$ を回帰できていることがわかる（最大で決定係数 0.99）．ここから，Transformer は明示的に計算の途中結果を保存するように訓練していなくても，暗黙的に数式の再帰的構造を捉え，途中結果を保存することができているといえる．今回の実験では訓練データをかなり少なくしたが，驚くべきことにすべての層から回帰できる結果となった．では，すべての層で計算途中結果の情報は同じように表現されているのだろうか．層ごとの違いを明らかにするため，次節では計算の流れをより仔細に調査する．

4.4　隠れ層の空間構造の分析

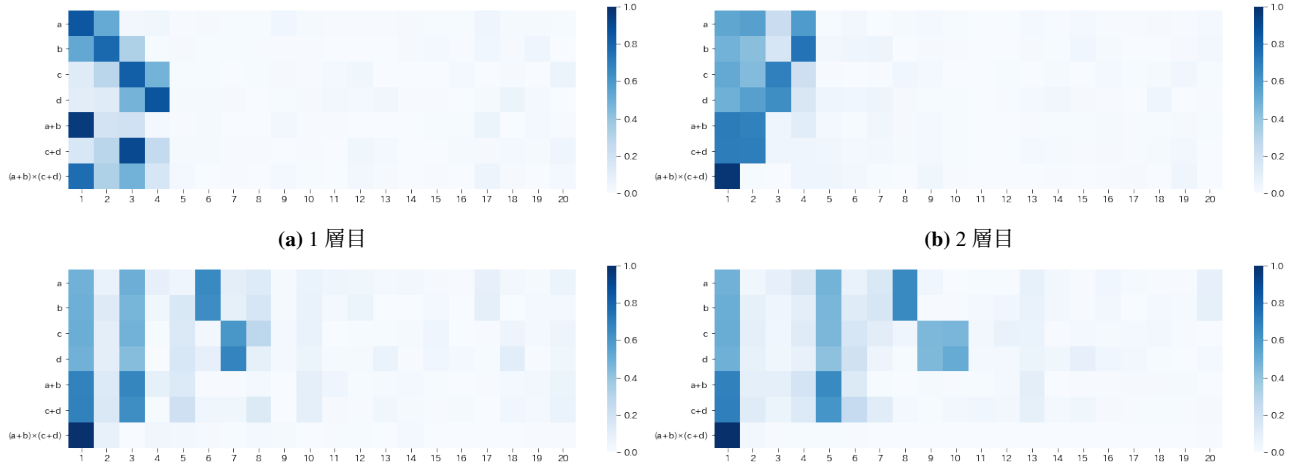
ここでは 4.3 節で示されたような再帰的構造把握能力がどのように達成されているのかについて，プロービングデータに対して内部表現の主成分分析（PCA）を行って検証する．

最終計算結果，計算途中結果と各成分の関係　本実験では 4.3 節で使用された 17,664 次元の結合済みベクトルに対して主成分数 20 で PCA を行い，PCA 後の各成分に対する計算中の途中結果や最終結果との相関関係を測定する．[3)] このうち，1 層目，2 層目，5 層目，6 層目における結果を図 4 に示す．[4)] 図 4 から，最終結果 $(a+b)\times(c+d)$ の情報は層が上がるにつれて第一主成分に集中することがわかる．一方で計算途中結果の情報を含む成分は特に上の層では分散して存在する．すなわち，計算の途中結果が分散的に表現されていることがわかる．例えば 2 層目に

2)　プロービングデータにおける数式の長さが 23 単語のため．

3)　主成分分析における符号は相対的なものであるため，相関係数は絶対値を取った．

4)　本文で載せていない層の結果については付録を参照されたい．



(a) 1 層目　　**(b)** 2 層目

(c) 5 層目　　**(d)** 6 層目

図 4: 層ごとの各主成分と計算結果の相関関係のヒートマップ．各セルは第 k 主成分（列）とある計算途中結果（行）との相関係数の絶対値を表す．

おいては第一主成分，第二主成分が $a+b$ と相関が強く，他の成分はほぼ相関がないが，5 層目や 6 層目ではある程度の相関を持つ成分の幅が広がっている．さらに，図 4(a)〜(d) の移り変わりを見ると，第一主成分以外で途中結果と強い相関を持つ成分が徐々に下位に移っていることがわかる．ここから，モデルにとって低い層では途中結果を重要な情報として持ち，上の層に行くにつれて途中結果が保存され続けてはいるものの重要でなくなっていることも示唆される．

主成分による計算途中結果プロービング　途中結果が分散的に表されているという観察結果を検証するため，4.3 節と同じ設定でのプロービングを再度行う．ただし，今回線形回帰器の入力となるのは PCA 後の上位 k 個の主成分である．例えば，$k=1$ の時は第一主成分の値のみから $(a+b)\times(c+d)$ 中の途中結果である $a+b$ の値を回帰することになる．このプロービングの結果を図 5 に示す．図 5 から，一般に k を増やすほど途中結果の値を回帰できるようになっていることが見て取れる．ここからも，モデルは計算の途中結果を複数の成分に**分散して**保存していることが裏付けられる．また，より詳細に見ていくと 1〜4 層目では 2 個以上の主成分を使用した時に性能が大きく上がっているが，5 層目，6 層目では $a+b$ の値と相関が高い成分が下位になっているため，5 層目では $k=2$，6 層目では $k=4$ での回帰スコアが急激に下がっている．これは図 4 と一致する結果である．

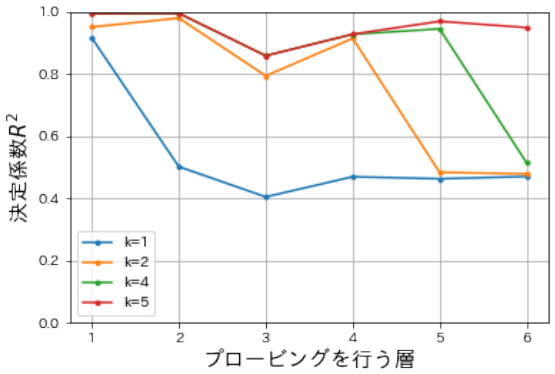


図 5: k 個の主成分による計算途中結果プロービングタスクの結果．上層においては寄与率が低い主成分を用いなければ解けない．

5　おわりに

本研究では計算の途中結果を保存することが求められるような数式を定義することで，Transformer の再帰的構造の把握能力の有無を調査した．その結果として，Transformer は内部的に計算の途中結果を教師なしで保存できていることがわかった．またこの時，計算の途中結果は分散的に保存されていることを示した．

今回は再帰性について検証を行ったが，数式の性質を用いることで処理の流れや構成性など別の能力を測ることも可能であると考えるため，今後の展望としたい．また，数式のような形式言語にあえて曖昧性を持たせるなどしてより自然言語に近い設定で再帰的構造の把握能力を調査する方向も考えられる．

謝辞

本研究は JST CREST JPMJCR20D2 及び JSPS 科研費 20K23314, 21K17814 の助成を受けたものです.

参考文献

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz 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**, Vol. 30. Curran Associates, Inc., 2017.

[2] Alessandro Raganato and Jörg Tiedemann. An analysis of encoder representations in transformer-based machine translation. In **Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP**, pp. 287–297, Brussels, Belgium, November 2018. Association for Computational Linguistics.

[3] Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does BERT learn about the structure of language? In **Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics**, pp. 3651–3657, Florence, Italy, July 2019. Association for Computational Linguistics.

[4] Ian Tenney, Dipanjan Das, and Ellie Pavlick. BERT rediscovers the classical NLP pipeline. In **Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics**, pp. 4593–4601, Florence, Italy, July 2019. Association for Computational Linguistics.

[5] Noam Chomsky. **Syntactic Structures**. Mouton and Co., The Hague, 1957.

[6] Brenden Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In Jennifer Dy and Andreas Krause, editors, **Proceedings of the 35th International Conference on Machine Learning**, Vol. 80 of **Proceedings of Machine Learning Research**, pp. 2873–2882. PMLR, 10–15 Jul 2018.

[7] Najoung Kim and Tal Linzen. COGS: A compositional generalization challenge based on semantic interpretation. In **Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)**, pp. 9087–9105, Online, November 2020. Association for Computational Linguistics.

[8] Tom McCoy, Ellie Pavlick, and Tal Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In **Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics**, pp. 3428–3448, Florence, Italy, July 2019. Association for Computational Linguistics.

[9] Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. Annotation artifacts in natural language inference data. In **Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)**, pp. 107–112, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[10] 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.

[11] Mor Geva, Ankit Gupta, and Jonathan Berant. Injecting numerical reasoning skills into language models. In **Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics**, pp. 946–958, Online, July 2020. Association for Computational Linguistics.

[12] Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. Do NLP models know numbers? probing numeracy in embeddings. In **Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)**, pp. 5307–5315, Hong Kong, China, November 2019. Association for Computational Linguistics.

[13] Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. Investigating the limitations of transformers with simple arithmetic tasks, 2021.

[14] Avijit Thawani, Jay Pujara, Filip Ilievski, and Pedro Szekely. Representing numbers in NLP: a survey and a vision. In **Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies**, pp. 644–656, Online, June 2021. Association for Computational Linguistics.

[15] Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. **Transactions of the Association for Computational Linguistics**, Vol. 4, pp. 521–535, 2016.

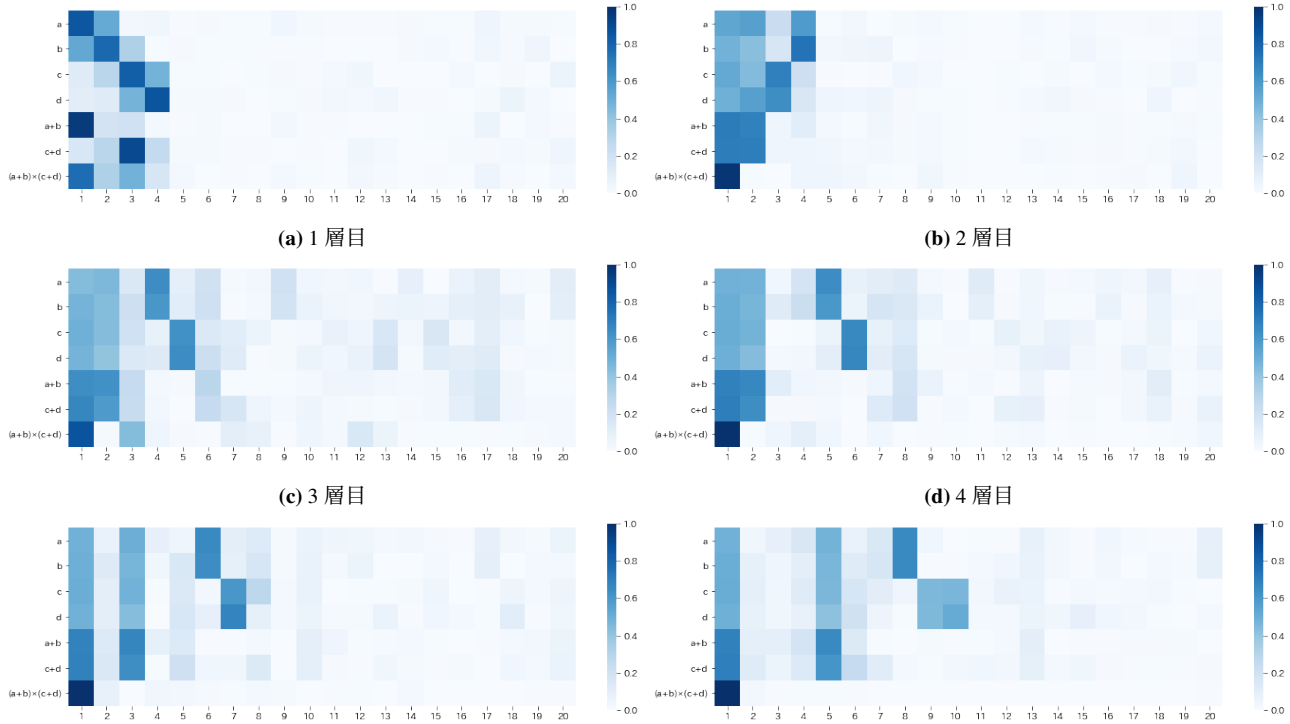
[16] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. **Neural Computation**, Vol. 9, No. 8, pp. 1735–1780, 11 1997.

[17] 岡本千尋, 内海慶, 持橋大地. 構文情報を陽に与えたときの lstm による内部表現について. 情報処理学会, 2018.

[18] Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. On the effect of dropping layers of pre-trained transformer models, 2021.

A　主成分分析の詳細な結果

4.4 節で載せられなかった 3 層目，4 層目も含めた主成分分析の結果を図 6 に示す．図 6 からも，やはり，第一主成分以外で途中結果と強い相関を持つ成分が徐々に下位に移っていることが見て取れる．



(a) 1 層目　**(b)** 2 層目

(c) 3 層目　**(d)** 4 層目

(e) 5 層目　**(f)** 6 層目

図 6: 全ての層における各主成分と計算結果の相関関係のヒートマップ．