統語情報は脳情報デコーディングに寄与するのか？

赤間美香　梶川康平　大関洋平
東京大学
{haruka-akama,kohei-kajikawa,oseki}@g.ecc.u-tokyo.ac.jp

概要

近年、大規模言語モデル（LLM）を活用し、脳活動データから言語表現を直接復元する脳情報デコーディングの手法が注目されている。一方、言語学や認知神経科学では、脳活動と言語表現の間に統語構造などの中間表現が存在することが広く想定されてきた。では、脳情報デコーディングにおいて、言語表現を直接復元するのではなく、統語情報のような中間表現を明示的に利用することで、デコーディング精度が向上する可能性はあるのだろうか。本研究では、脳波データを用いて言語生成を行う BrainLLM を活用し、品詞情報を中間表現として取り入れることで、デコーディング精度が向上するのか検証した。その結果、統語情報の考慮がデコーダーの性能向上に寄与する可能性が示唆された。

1　はじめに

人間は言語を使って思考したり他者とコミュニケーションをするが、こうした言語活動は、すべて脳内の神経活動に基づいている。こうした神経活動の仕組みを解明することは、言語や人間の仕組みの科学的理解に寄与するだけでなく、失語症の治療やリハビリテーションなどといった医療分野への応用も期待される。脳情報デコーディングは、脳波や血行動態などの脳活動データから言語表現を復元する技術であり、言語の神経基盤の解明や医療への応用に向けた重要な一歩となる [1]。近年、大規模言語モデル（LLM）を活用し、脳活動データをもとに言語生成を行う BrainLLM [2] が提案されるなど、LLM を利用した脳情報デコーディングが注目されている。

一方、言語学や認知神経科学では、脳活動と実際の言語表現の間に統語構造などの中間的な言語表現が存在することが想定されてきた。たとえば、言語学者である Jackendoff は、音韻、統語、意味の 3 つの構造が相互に関連しながら言語表現が構築される理論を提唱している [3]。言語表現を音韻、統語、意味の 3 つの部門で分離して考え、それぞれ、音韻構造は統語構造に渡され、そして最終的に意味（概念）構造に至ることで構築されているという。また、統語と意味の神経基盤が分離していることも神経言語学の研究から示唆されている [4]。

では、脳情報デコーディングにおいて、脳活動データから直接言語表現を復元するのではなく、統語情報のような中間表現を利用することで、デコーディング精度を向上させることが可能なのだろうか。言い換えると、脳活動データと言語表現の乖離を、中間表現で埋めることにより、デコーディングの質が向上する可能性はあるのだろうか。

本研究では、脳波データを用いた BrainLLM を活用し、品詞情報を統語的な中間表現として取り入れることがデコーディングに与える効果を検証した。その結果、統語情報を明示的に考慮することで、デコーダーの性能が一定程度向上する傾向が示唆された。

2　モデルの提案

BrainLLM　脳活動データから意図されている言語表現を再構築する試みは、事前に生成された言語候補を、脳からデコードされた意味表現とどれだけ一致するかに基づいて選択する分類の枠組み内で検証されてきた [5, 6, 7]。ここで、脳情報から最も可能性の高い意味の候補が選択されるのであって、脳情報が言語生成プロセスに直接使用されることはなかった。そこで、Ye らによって提案されたのが、脳記録からデコードされた意味表現を言語生成段階に直接組み込む BrainLLM である [2]。BrainLLM は脳記録を言語生成段階へ直接使用する可能性を提示し、以前のアプローチよりも BrainLLM のアプローチの方が優れていることを示した。

BrainLLM_with_POS　本研究では統語情報がデコーダに有効かを調べるために BrainLLM に POS を明示的に与える BrainLLM_with_POS を提案する。

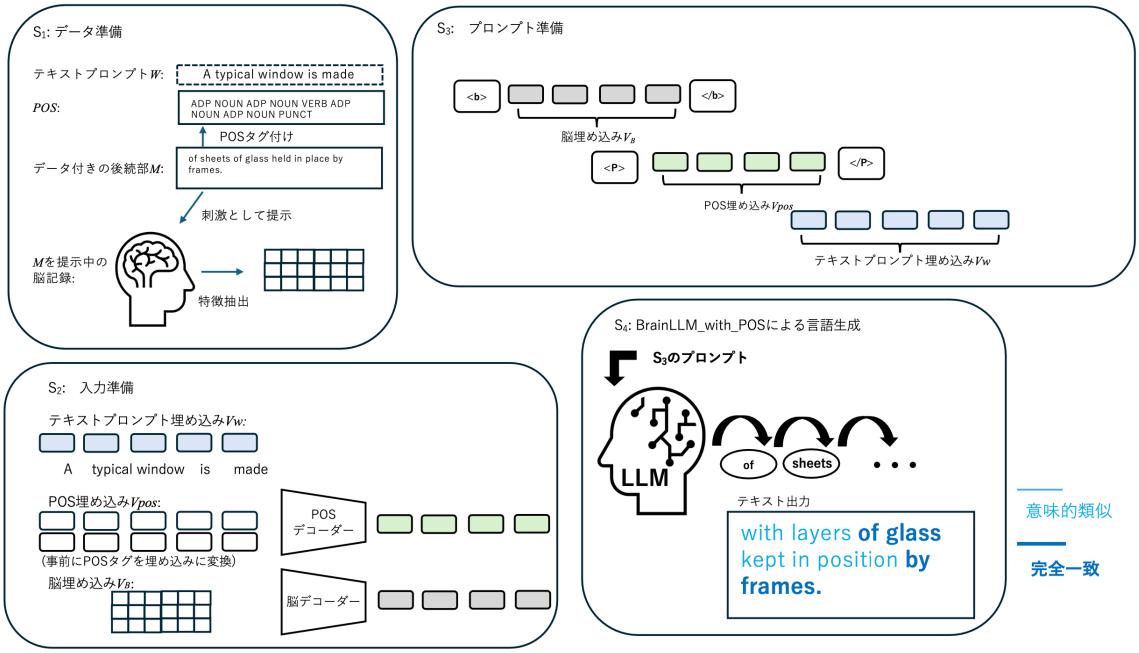

図 1　BrainLLM_with_POS による言語生成過程の概要

基盤モデルとして、Huggingface で公開されている GPT-2 [8][1] を使用した。

BrainLLM_with_POS は BrainLLM を参考に 4 つ主要なステップで構成されている（図 1）。

1. 脳活動データを収集し特徴を抽出することと並行して生成ターゲットである知覚データ付きの後続部 (perceived continuatuion) に POS タグを付加。
2. 脳デコーダが脳活動データから埋め込みを学習し、POS デコーダーが（事前に POS タグから埋め込みに変換した）POS 埋め込みを学習。
3. 脳と品詞情報（POS）とテキストの 3 つからプロンプトを構築。
4. プロンプトを LLM に入力して、自動回帰的に言語を生成。

3　実験方法

3.1　タスク定式化

まず、Pereira データセット [7] の各文章を 3 等分し、最初の 1/3 をテキストプロンプト、2/3 を知覚データ付きの後続部 (perceived continuation) M とするデータサンプル、または、最初の 2/3 をテキストプロンプト、最後の 1/3 を知覚データ付きの後続部 M とするデータサンプルを作る。次に、実験参加者が M を刺激として知覚したときの脳活動記録を B とする。さらに、M を POS 埋め込みにしたものを P とする。このとき、モデルに、W から B と P を利用して文字列の後半部である $M = \{m_1, m_2, \cdots, m_k\}$ を予測させることがこの実験におけるタスクである。言い換えれば、言語生成タスクとして、脳活動データ B と POS 埋め込み P とテキストプロンプト W を入力として、知覚データ付きの後続部 M を 1 トークンずつ生成する自己回帰関数 F を学習させることであるともいえる。このプロセスは以下のように定式化される：

$$\hat{m}_i = F(\{w_1, \ldots, w_n, \hat{m}_1, \ldots, \hat{m}_{i-1}\}, B, P; \Theta) \quad (1)$$

ここで、$w_1, \ldots, w_n$ はテキストプロンプト W におけるトークン列、$\hat{m}_i$ はモデルが生成する i 番目のトークン、Θ はモデルのパラメータを指す。

3.2　プロンプト準備

最初に、テキストプロンプトを直接 LLM の埋め込み層 f_w に入力し、トークンを潜在ベクトル $V_W = \{v_W^1, \ldots, v_W^n\} \in \mathbb{R}^{n \times d}$ に変換する。ここで、n はトークンの数、d は埋め込みサイズを表す。次に、脳活動データを同じ潜在空間に埋め込むための脳デコーダ f_b を設計する。このデコーダは、各 $b_i \in B$ を空間 $\mathbb{R}^d$ に埋め込むもので、次のように定式化される：

$$v_B^i = f_b(b_i) \quad (2)$$

1)　https://huggingface.co/gpt2

さらに、POS タグを同じ潜在空間に埋め込むための POS タグデコーダ f_{pos} を設計する。このデコーダも各 $\text{pos}_i \in \text{POS}$ を空間 $\mathbb{R}^d$ に埋め込むもので、次のように定式化される：

$$v^i_{\text{POS}} = f_{\text{pos}}(pos_i) \tag{3}$$

最後に、脳埋め込み V_B と POS 埋め込み V_{POS} をテキスト埋め込み V_W と連結し、LLM がテキストに加えて、脳活動データと POS という 3 つの情報を統一された表現で認識できるようにする。これら 3 つの情報を効果的に区別するために、脳埋め込みには特殊トークン ⟨brain⟩ と ⟨/brain⟩を導入し、POS 埋め込みには特殊トークン ⟨POS⟩ と ⟨/POS⟩を用いて、それぞれの埋め込みの開始と終了を示す。これらの特殊トークンは、それぞれ 1 次元のランダム初期化されたベクトル $v_{\text{<brain>}}$ と $v_{\text{</brain>}}$、$v_{\text{<POS>}}$ と $v_{\text{</POS>}}$ として定義される。これらのベクトルは、LLM のトークン埋め込みと同じ次元 d を持っている。その結果、入力列 I は以下のように定式化される：

$$\begin{aligned} I = \{&v_{\langle\text{brain}\rangle}, v^1_B, \ldots, v^t_B, v_{\langle/\text{brain}\rangle}, \\ &v_{\langle\text{POS}\rangle}, v^1_{\text{POS}}, \ldots, v^k_{\text{POS}}, v_{\langle/\text{POS}\rangle}, \\ &v^1_W, \ldots, v^n_W\}. \end{aligned} \tag{4}$$

3.3 脳デコーダと POS デコーダ

脳デコーダ f_b は深層ニューラルネットワークであり、脳活動データ $B\{b_1, \cdots, b_t\} \in \mathbb{R}^{t\times c}$ を入力として受け取り、脳埋め込み $V_B = \{v^1_B, \ldots, v^t_B\} \in \mathbb{R}^{t\times d}$ を出力する。d は LLM の埋め込みサイズである。BOLD 信号の i-番目の時系列データ b_i に対応する埋め込みベクトルは、以下のように脳デコーダ fb によって計算される。このとき、p は BOLD 信号の収集時の時間順序を表す位置埋め込みを指す：

$$v^i_B = f_b(b_i) = f_m(p_i + b_i) \tag{5}$$

出力埋め込みベクトル v^i_B は LLM の埋め込みサイズに合わせた次元を持つ。POS タグ $POS\{\text{POS}^1, \cdots, \text{POS}^k\}$ は PyTorch の ‘torch.nn.Embedding‘によって 1000 次元の POS 埋め込み $V_{\text{POS}} = \{v_{\text{POS}}{}^1, \ldots, v^k_{\text{POS}}\} \in \mathbb{R}^{k\times 1000}$ に変換され、POS デコーダに渡される。POS デコーダ f_{pos} は深層ニューラルネットワークであり、1000 次元の POS 埋め込み $V_{\text{POS}} = \{v_{\text{POS}}{}^1, \ldots, v^k_{\text{POS}}\} \in \mathbb{R}^{k\times 1000}$ を入力として受け取り、POS 埋め込み $V_{\text{POS}} = \{v_{\text{POS}}{}^1, \ldots, v^k_{\text{POS}}\} \in \mathbb{R}^{k\times d}$ を出力する。POS タグの i-番目の時系列データ pos_i に対応する埋め込みベクトルは、以下のように POS デコーダ f_{pos} によって計算される：

$$v^i_{\text{POS}} = f_{\text{POS}}(\text{pos}_i) \tag{6}$$

出力埋め込みベクトル v^i_{POS} は LLM の埋め込みサイズに合わせた次元を持ち、テキストと脳情報と組み合わせて入力を構成するために使用される。

3.4 トレーニング

このモデルのトレーニングは、[9] で提案されたプロンプトチューニング技術に着想を得て、次の 2 つのステップで構成されている：

1. ウォームアップステップでは埋め込みの分布をテキストトークン埋め込みの分布に整列させることを目的にトレーニングを行う。ウォームアップステップのトレーニングには、平均二乗誤差（MSE）損失を採用した。
2. メイントレーニングステップではウォームアップが完了した後、脳埋め込みと POS 埋め込みがテキストの埋め込みと適切に統合されるようにモデル全体をトレーニングすることを目的にトレーニングする。オプティマイザーは Adam [10] を使用し、学習率を 1×10^{-4}、バッチサイズを 8 に設定した。ウォームアップステップのトレーニングは 10 エポックで停止した。

学習の目標は知覚データ付きの後続部の生成確率を最大化することである。メイントレーニングステップでは、LLM の固有の知識を保持しつつ、「プロンプトチューニング」技術 [11] を用いて少数のデータサンプルから有用な情報を学習する。この技術では、LLM のパラメータを変更せずに、代わりに入力表現（$\Theta_{fb}, \Theta_{\text{fpos}}, \Theta_{sp(brain)}, \Theta_{sp(pos)}$）のみを微調整する。脳デコーダは人間の脳活動データを解読し、POS デコーダは POS 情報を解読することで、LLM が知覚データ付きの後続部に近い出力を生成するように導く。

3.5 評価指数

本研究では、[2] に倣い、言語類似性指標として、BLEU [12]、ROUGE [13]、および WER [12] を採用した。BLEU は、生成された出力と知覚データ付きの後続部の n-gram を比較し、一致数をカウントする ROUGE は、n-gram のオーバーラップを計算する一連の指標である。本研究では、ともに $n = 1$ で評価

データセット	LLM Backbone	モデル	BLUE-1(↑)	ROUGE-1(↑)	WER(↓)
Pereira's	GPT-2 (117M)	LLM	0.105937	0.109504	0.951497
		LLM_with_perPOS	0.180369	0.171204	0.887677
		LLM_with_POS	0.180695	0.170119	0.886119
		perBrainLLM	0.150302	0.141861	0.916319
		perBrainLLM_with_perPOS	0.161956	0.156028	0.906577
		perBrainLLM_with_POS	0.165693	0.157083	0.901276
		BrainLLM	0.156483	0.148162	0.907579
		BrainLLM_with_perPOS	0.162293	0.158180	0.907576
		BrainLLM_with_POS	0.164124	0.158029	0.905006

表 1　モデルごとの言語生成能力の結果

する。WER は、生成された出力と知覚データ付きの後続部の間のエラー（置換、削除、挿入）の総数の比率を計算する。一般に、BLEU および ROUGE のスコアは高いほど言語類似性が高いことを意味する一方、WER のスコアは低いほど、言語類似性が高いことを意味する。

3.6　データセットと前処理

BrainLLM_with_POS は公開 fMRI データセットの Pereira データセット [7] でテストする。Pereira データセットは、Wikipedia 形式である 627 の文章を読んでいる間の 5 人の被験者の BOLD 信号を収集したものである。被験者には各文章が重複なく提示された。本研究では Ye らによって前処理された脳活動データ（BOLD 特徴に対して次元削減を適用し、全てのデータセットで次元数が $c = 1000$ に圧縮）を用いる。データサンプルは約 3:1:1 の比率で、訓練セット、検証セット、テストセットに分割した。分割された訓練セット、テストセット、検証セットにおいて知覚データ付きの後続部や脳活動データは重複しない。また、各データへの POS は、`Berkeley Neural Parser`[2] [14] を用いて付与した。

4　結果・考察

BrainLLM_with_POS の生成性能を 8 つのコントロールモデルと比較した。LLM は脳活動データを使用しない標準的な LLM であり、テキストプロンプトのみを使用して言語を生成する。PerBrainLLM は BrainLLM と同じ手法を使用するが、脳データがランダムに並び替えられている。この並び替えにより、脳活動データと知覚データ付きの継続部との対応が乱され BrainLLM に対するコントロールモデルとして機能する。同様に、PerPOS は POS と同じ手法を使用するが、POS 情報がランダムに並び替えられている。この並び替えにより、POS 情報と知覚データ付きの後続部との対応が乱され POS に対するコントロールとして機能する。実験の結果を表 1 にまとめる。結果として、BrainLLM よりも BrainLLM_with_POS の方が 3 つの言語類似性指標すべてにおいて性能が良いことがわかった。しかし、3 つのモデルにわたって perPOS と POS の間でほとんど性能が変わらなかった。また、LLM に脳活動データを与えるよりも POS を与えた方が性能が大きく向上した。性能が最も良かったのは LLM_perPOS もしくは LLM_POS であった。

5　おわりに

実験の結果、統語情報を明示的に取り入れることで脳情報デコーディング性能が高くなる傾向が観察された。この結果から統語情報はデコーダの性能改善に効果的であることが示唆される。しかし、一方で perPOS と POS の間ではほとんど性能に差がないことも示されており、この点については今後十分な検討が必要である。また、統語情報は BrainLLM よりも LLM に与えた方が効果的であることも示された。今後は、今回の研究結果を踏まえ、POS 以外の中間表現についても、デコーダの生成能力の改善に寄与するか調べていきたい。

2）https://github.com/nikitakit/self-attentive-parser

謝辞

本研究は、JST さきがけ JPMJPR21C2 および JSPS 科研費 24H00087 の支援を受けたものです。

参考文献

[1] Susan Fager, David R Beukelman, Melanie Fried-Oken, Tom Jakobs, and John Baker. Access interface strategies. **Assistive Technology**, Vol. 24, No. 1, pp. 25–33, 2012.

[2] Ziyi Ye, Qingyao Ai, Yiqun Liu, Maarten de Rijke, Min Zhang, Christina Lioma, and Tuukka Ruotsalo. Language generation from brain recordings, 2024.

[3] Ray S. Jackendoff. **Semantics and cognition**. No. 8 in Current studies in linguistics series. MIT Press, 1983.

[4] Dietrich Klakow and Jochen Peters. Testing the correlation of word error rate and perplexity. **Speech Communication**, Vol. 38, No. 1, pp. 19–28, 2002.

[5] Tom M. Mitchell, Svetlana V. Shinkareva, Andrew Carlson, Kai-Min Chang, Vicente L. Malave, Robert A. Mason, and Marcel Adam Just. Predicting human brain activity associated with the meanings of nouns. **Science**, Vol. 320, No. 5880, pp. 1191–1195, 2008.

[6] Xiaomei Pei, Dennis L Barbour, Eric C Leuthardt, and Gerwin Schalk. Decoding vowels and consonants in spoken and imagined words using electrocorticographic signals in humans. **Journal of neural engineering**, Vol. 8, No. 4, p. 046028, 2011.

[7] Francisco Pereira, Bin Lou, Brianna Pritchett, Samuel Ritter, Samuel J Gershman, Nancy Kanwisher, Matthew Botvinick, and Evelina Fedorenko. Toward a universal decoder of linguistic meaning from brain activation. **Nature communications**, Vol. 9, No. 1, p. 963, 2018.

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

[9] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. **ACM Computing Surveys**, Vol. 55, No. 9, pp. 1–35, 2023.

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. **CoRR**, Vol. abs/1412.6980, , 2014.

[11] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. Gpt understands, too. **AI Open**, Vol. 5, pp. 208–215, 2024.

[12] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In Pierre Isabelle, Eugene Charniak, and Dekang Lin, editors, **Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics**, pp. 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

[13] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In **Text Summarization Branches Out**, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[14] Nikita Kitaev, Steven Cao, and Dan Klein. Multilingual constituency parsing with self-attention and pre-training. In **Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics**, pp. 3499–3505, Florence, Italy, July 2019. Association for Computational Linguistics.