Transformer における Attention 機構の仕組み

早稲田大学 理工学術院 村田 昇，鈴木 詠斗

Noboru Murata, Eito Suzuki
Waseda University, Faculty of Science and Engineering

大規模言語モデル (Large Language Models; LLM) の急速な進展は，自然言語処理 (Natural Language Processing; NLP) の枠組みそのものを根本的に変えようとしている．この変革において重要な要素となっている技術が Transformer である．Transformer モデルは，Multi-Head Attention (MHA) と Multi-Layer Perceptron (MLP) の組み合わせで構成され，これらの層が連携して高度な特徴表現を生成する．MHA はトークン間の複雑な関係を捉え，MLP は非線形性を導入することで表現力を向上させ，両者が一体となってモデルの強力な文脈理解と文章生成の機能を支えている．Transformer の内部処理を理解するためにさまざまな方法が提案されているが，その一つとして Integrated Gradients (IG) と Layer-wise Relevance Propagation (LRP) を組み合わせた解釈手法を紹介する．

1 はじめに

統計的な自然言語処理 (Natural Language Processing; NLP) の根底にある目的は，実は驚くほど単純で不変である．それは，「与えられた文脈 C のもとで，次にどの語句 w が出現するか」という条件付確率 $P(w|C)$ をいかに正確に推定するかということである．初期の統計モデルから現代の大規模言語モデル (Large Language Model; LLM) に至るまで，この目的自体はなんら変わっていない．しかし，天文学的な数になる語句の組み合わせをどのような形式で記憶し，計算するかという方法論において，自然言語処理の歴史には興味深い変革が繰り返されてきた．本稿では，その変遷を概観し，現在の LLM の基盤である Transformer [1] がもたらした情報処理の革新と，その内部を可視化する重要性についてまとめる．

初期の統計的な NLP は，単語の組み合わせを離散的なパタンとして捉え，巨大な集計表 (テーブル) で管理する手法が主流であった．n-gram モデルは出現頻度を数え上げた確率表を用いたが，文脈を長くしようとすると表のサイズが指数関数的に増大する問題に直面し，一度も見たことのない組み合わせには対応できないという限界があった．例えば隠れマルコフモデル (Hidden Markov Model; HMM) のように潜在的な構造を導入して，表の自由度を制御する試みもあるが，本質的には依然として離散的な確率表を記述する枠組みにとどまっていた．

変革の契機は，単語を離散的な記号ではなく連続的な数値ベクトルとして扱う分散表現 (distributed representation) の導入であろう．単語の意味をベクトルとして捉える分散表現は Milkov らの word2vec [2] にはじまり，単語の類似性の比較や演算が可能であることが示された．続いて文書全体を単語の平均ベクトルで表現する doc2vec [3] も登場し，文書レベルでの理解に分散表現を用いる可能性が生まれた．更に，確率分布の保持形式を「離散的な表」から，ニューラルネットワークを利用した「連続関数」へと転換することにより，劇的な変化が起きた．ベクトル空間上で似た意味の単語

を近くに配置するという幾何学的性質を持たせることで，モデルは学習データにない未知の組み合わせであっても近傍情報を活用して確率を補間する，ある種の汎化能力を獲得した．これにより，巨大な確率表の成分数よりも大幅に少ないパラメタで，言語の確率的構造を効率的に記述することが可能となった．つまり，NLP の進化とは，「大規模な確率表をいかにして低次元のパラメタに圧縮し，かつ汎化性能の高い滑らかな関数として表現するか」という戦いの歴史と考えることができる．

分散表現とニューラルネットワークの利用により条件付確率の推定精度は飛躍的に向上したが，文章の順序や構造を扱うために導入された LSTM (Long Short-Term Memory) [4] などの再帰型ネットワーク (Recurrent Neural Network; RNN) は，長文処理における忘却問題と逐次処理による計算コストの増大という課題を抱えていた．2017 年に提唱された Transformer [1] は，この制約を根本的に撤廃し，自然言語処理に新たな情報処理様式をもたらした．

Transformer の最大の特徴は Attention 機構にある．文中のすべての単語が互いに直接関係を持つ全結合処理を導入したことで，文中の単語の位置に関らず単語同士の重要な情報の関連付けを行うことができ，逐次処理による忘却問題が解消された．また，逐次処理を排したため大規模な並列計算が可能となり，現代の LLM の飛躍的な性能向上に繋ったと考えられる．

しかしながら Transformer の面白さは，単に計算効率が高いことだけではない．情報処理の観点から見た Transformer は，入力された単語ベクトルの集合 (点群) を，複数の Attention 機構 (Multi Head Attention; MHA) と多層パーセプトロン (Multi Layer Perceptron; MLP) を通じて，次の語の予測に最適な情報を持つ特徴ベクトルの点群へと動的に再構成するベクトル集合の変換器である．点群を確率分布の粒子近似による表現と捉えれば，入力された単語ベクトルの分布から情報処理に適した特徴ベクトルの分布へと，確率的な構造を変換する過程と考えることもできる．複数の Transformer 層での処理を通して情報の重み付けや特徴の統合が逐次行われ，入力時点ではある種乱雑に散らばった言語データの点群が，文脈の意味を捉えながら新しい点群へと整理されていく．この点群の変換過程は，従来のパターンマッチングや固定的な文脈ベクトルの処理とは一線を画す，データ集合そのものを操作する新しい機械学習の形態と捉えることができる．

Transformer の登場により，大規模な文脈理解と高精度な文章の生成が可能となり，LLM の登場を可能にした．これらのモデルは，単なる言語処理を超えて，エージェント化やマルチモーダル処理 (画像 + 言語 + 音声) へと進化しており，今後は人間の思考に近い自律的な言語処理までもが期待されている．

2 学習機械の説明可能性

前節で述べたように，現代の大規模言語モデル (LLM) は，入力の点群を層ごとに動的に再構成する高度な変換器として機能している．数千億から数兆規模のパラメタが複雑に絡み合い，高次元の非線形変換を繰り返し適用するこの情報処理機構は，驚異的な文脈理解と文章生成能力を発揮する一方で，その内部でどのような計算が行われ，どの情報が根拠となって出力が導出されているのかを人間が追跡することは極めて困難である．この現象は「ブラックボックス問題」として知られ，その性能向上と相まって，その予測の信頼性・安全性・責任性を確保する上での大きな課題となっている．

特に，LLM が提示する回答の背後にある推論過程や，特定の出力を決定づけた語句の寄与を定量的に特定できない状況は，医療・法務・教育など高信頼性が要求される分野での利用を阻む要因となっている．モデルが高度化した結果，内部の情報処理は人間の直感や従来の理論の枠組みを超えた複雑さを帯びるようになった．したがって，単に高い精度を維持するだけでなく，その振る舞いの根拠を透明化し，必要に応じて制御・検証可能な枠組みを構築することは，現代の人工知能研究において不可欠な課題と考えられている．

ブラックボックス内部の可視化と解釈可能性の向上を目指し，多様な分析手法が提案され，その

方法論の整理も進んでいる (例えば [5], [6], [7] などを参照)．学習後の LLM を対象とした事後解析的な方法においては，以下のような考え方が主に用いられる:

1. 因子属性分析: モデルの出力に対する入力の寄与を計測する手法．勾配ベースの手法 (Saliency Maps など) に加え，Integrated Gradients (IG) や Layer-wise Relevance Propagation (LRP) は，非線形関数の出力変化を各入力成分への定量的な寄与として分解し，情報の伝播経路を追跡する．これらの手法は，モデルがどの入力に注目して判断を下したかを数値的に提示する点で実用性が高い．

2. 代替モデル分析: 複雑な学習器の特定の入力まわりの局所的な挙動を，解釈可能な簡易モデル (線形モデル，決定木，規則ベースシステムなど) で近似し，その代替モデルを通じて元のモデルの意思決定過程を説明する手法．

3. ネットワーク分析: ニューラルネットのようなネットワーク構造を持つ情報処理機構の内部の活性化パターン，重み行列，または潜在表現を直接解析し，特定の素子や部分ネットワークが担う機能 (文法構造の抽出，事実知識の参照，推論ステップの逐次実行など) を特定しようとする手法．回路分析 (Circuit Analysis) や活性化ベクトルの幾何学的解析が注目を集めている．

これらの手法はそれぞれ異なる視点からモデルの解釈を試みるが，単独で完結する説明を与えることは基本的に困難である．例えば，Attention Weight (Attention 機構で計算される重み) をそのまま注目度や因果的根拠として解釈することは，必ずしもモデルの情報処理経路と一致しないことが実験的に示されており，指標の選択や可視化の手法に依存して誤解を招く可能性がある．したがって，複数の手法を相互補完的に組み合わせ，情報の伝播・変換・統合の変換過程を多角的に検証する枠組みを構築する必要がある．

本稿では，こうした解釈可能性の課題に対し，LLM，特に Transformer の情報処理を因子属性分析の視点から捉える方法を提示する．高次元の潜在表現を意味的・構文的・文脈的・推論的などの潜在因子に分解し，各因子が情報の変換過程においてどのように活性化・抑制・統合されていくかを追跡することが最終的な目的である．このような分析の有効性は，単にブラックボックスを開くことだけでなく，モデルがどのように文脈を構造化し，知識を階層的に統合しているかを定量的に記述できる点にある．これにより，LLM の振る舞いを人間が理解可能な因子の相互作用として捉え，モデルの改善・制御・信頼性評価に直接結びつくことが期待される．特に，層ごとの情報処理過程が局所的な文脈統合から大域的な意味統合へと移行していく様子を，因子の活性化パターンとして追跡することで，Transformer の動的な点群変換過程を定量化できる可能性が生まれる．

次節では，分析手法を適用するための理論的基盤として，Transformer の基本的な構成要素である Multi-Head Attention (MHA) と Multi-Layer Perceptron (MLP) の情報処理機構を記述する．これらの層がどのように連携して特徴表現を生成し，情報の流れを形成しているかを明確化した上で，分析の考え方を示す．

3 Transformer の構造

解くべき課題に応じて様々な拡張があるが，Transformer [1] の基本的な機能は以下の 2 つである:

- 文章をその文脈を反映した有限次元のベクトルに変換すること，
- 文脈に基づいて次の語句を予測すること．

前者は符号器 (encoder) と呼ばれる部分が担い，文章の意味を数値ベクトルとして表現することを可能にする．直感的には「概念空間」を構築していると捉えることができ，文書のクラスタリング

や類似文の検索などに応用される．後者は復号器 (decoder) と呼ばれる部分が担い，文の構造・文法・意味を反映した自然な文の生成を行う．言語モデルの本質的な機能でもあり，翻訳，要約，会話生成などに応用される．復号器は文脈を理解するために符号器を利用して直前までに入力された文章をベクトル化し，このベクトル表現と候補の単語のベクトルの適合度を数量化して条件付確率を計算している．

以下では符号器について説明する．Transformer への入力文は，まずトークナイザ (tokenizer) と呼ばれる機構により形態素などのトークン (token) に分割され，適切に学習された辞書を用いて初期のベクトルが与えられる．入力されたトークン列のベクトル集合は Multi-Head Attention (MHA) 層と Multi-Layer Perceptron (MLP) 層からなる Transformer 層で繰り返し変換を受け，トークン間の相互関係を反映したベクトル集合に変換される (Fig.1 参照)．MHA 層と MLP 層は相互に連携して高度な特徴表現を生成し，モデルの表現力と汎化性能を向上させている．文脈情報を反映した文章のベクトル表現は，一般に多数の Transformer 層で変換され得られた最終層のベクトル集合の重み付き平均として与えられる．

3.1 位置符号化と初期表現の構成

Attention 機構はトークン間の関係性を並列に評価する性質上，入力された系列の順序情報を自発的に捉えることができない．この課題を解決するため，各トークンの初期ベクトルにその系列内での位置情報を加える位置符号化 (Positional Encoding) が導入されている．位置符号化は，正弦・余弦関数を用いた固定ベクトルや，学習可能な埋め込みベクトルなど複数の実装が存在するが，いずれもトークン i の初期ベクトル表現 $\boldsymbol{o}_i$ (文脈に無関係) に文章内のトークンの位置情報 $\boldsymbol{p}_i$ を加算することで，トークンの系列構造をベクトル空間に明示的に導入する:

$$\boldsymbol{x}_i = \boldsymbol{o}_i + \boldsymbol{p}_i, \quad \boldsymbol{x}_i, \boldsymbol{o}_i, \boldsymbol{p}_i \in \mathbb{R}^p, \quad i \in \mathcal{I} = \{1, \ldots, I\}, \ (i \text{ はトークンを区別する添字}).$$

これにより，Transformer は順序を無視した点群ではなく，文脈的順序を保持した点群として入力を扱うことが可能となる．この初期表現は，後述する MHA と MLP が文脈依存の特徴を再構成するための基盤となる．

3.2 Multi-Head Attention

Transformer の中心的な構成要素である Attention 機構は，トークン間の依存関係を評価し，文脈の重要な情報を強調する仕組みである．入力されたトークン列 (ベクトル集合) を $\boldsymbol{x}_i \in \mathbb{R}^p$, $i \in \mathcal{I}$ と

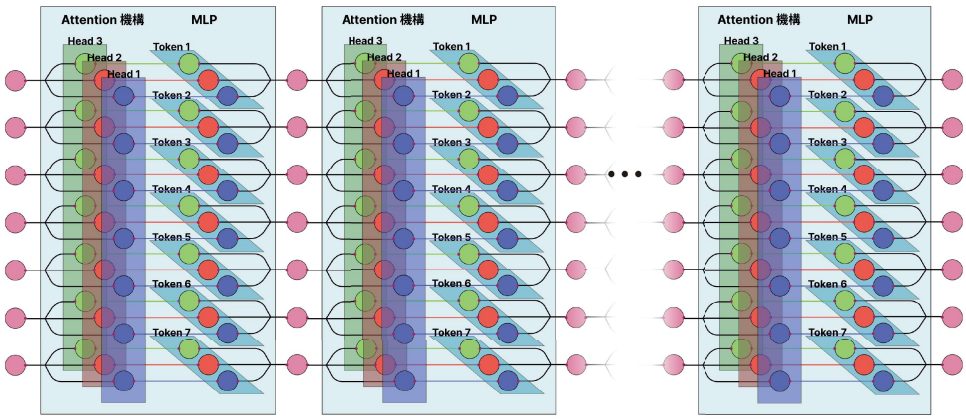

Figure 1: Transformer の基本構造．左から入力されたトークン列は，複数の MHA 層と MLP 層で文脈を反映したベクトル集合に変換される．

すると，Attention ブロックではまず各トークンごとに Query，Key，Value と呼ばれる 3 種類のベクトルに変換される:

$$
\begin{aligned}
\boldsymbol{q}_i &= W_Q \boldsymbol{x}_i + \boldsymbol{b}_Q, \quad W_Q \in \mathbb{R}^{q\times p}, \boldsymbol{b}_Q \in \mathbb{R}^q, \\
\boldsymbol{k}_i &= W_K \boldsymbol{x}_i + \boldsymbol{b}_K, \quad W_K \in \mathbb{R}^{q\times p}, \boldsymbol{b}_K \in \mathbb{R}^q, \\
\boldsymbol{v}_i &= W_V \boldsymbol{x}_i + \boldsymbol{b}_V, \quad W_V \in \mathbb{R}^{r\times p}, \boldsymbol{b}_V \in \mathbb{R}^r.
\end{aligned}
$$

Query，Key，Value への変換は非線形変換に一般化することもできるが，線形変換を用いるのは主に GPU を効率的に利用するための実装上の理由からである．トークン i のトークン j に対する Attention Weight w_{ij} は，Query $\boldsymbol{q}_i$ と Key $\boldsymbol{k}_j$ の内積を用いて計算され，$(w_{i1}, \ldots, w_{iI})$ が確率ベクトルとなるように SoftMax 関数により正規化される:

$$
\begin{aligned}
\tilde{w}_{ij} &= \boldsymbol{q}_i \cdot \boldsymbol{k}_j, \quad i, j \in \mathcal{I} = \{1, \ldots, I\}, \\
\{w_{ij}, i, j \in \mathcal{I}\} &= \mathrm{SoftMax}(\{\tilde{w}_{ij}, i, j \in \mathcal{I}\}) \\
&= \frac{\exp(c\tilde{w}_{ij})}{\sum_{j\in\mathcal{I}} \exp(c\tilde{w}_{ij})}, \quad w_{ij} \geq 0, \ \sum_{j\in\mathcal{I}} w_{ij} = 1, \ (c \text{ は定数}).
\end{aligned}
$$

各 Weight は 0 から 1 の範囲となり，関連性の高い要素に高い重みが割り当てられる．この Attention Weight を Value に適用し，重み付き和を計算することで他のトークンの情報を取り込んだ新しいベクトル表現を得る:

$$
\boldsymbol{u}_i = \sum_{j\in\mathcal{I}} w_{ij} \boldsymbol{v}_j, \ i \in \mathcal{I}
$$

結果的に Attention ブロックではベクトル集合 $\{\boldsymbol{x}_i\}$ からベクトル集合 $\{\boldsymbol{u}_i\}$ への変換が行われることになる:

$$
\{\boldsymbol{u}_i, i \in \mathcal{I}\} = \mathrm{ATT}(\{\boldsymbol{x}_i, i \in \mathcal{I}\}).
$$

さて 1 つの Attention ブロックは，いわば 1 つの視点からの情報抽出に限られるため，複雑な依存関係を捉えるのは困難である．これを克服するために MHA 層では複数の Attention ブロック (ヘッドと呼ばれる) での計算が並列に実行され，それぞれのヘッドが異なる視点で情報を抽出する:

$$
\{\boldsymbol{u}_i^{(h)}, i \in \mathcal{I}, h \in \mathcal{H}\} = \mathrm{MHA}\left(\{\boldsymbol{x}_i, i \in \mathcal{I}\}\right), \quad \mathcal{H} = \{1, \ldots, H\} \ (h \text{ はヘッドを区別する添字}).
$$

これにより，モデルは文法的関係，意味的関係，時間的関係など，多様な依存性を同時に捉えることが可能となる．

Attention Weight w_{ij} はトークン間の有向グラフのエッジの重みと解釈でき，後述する Layer-wise Relevance Propagation (LRP) や Integrated Gradients (IG) の解析対象となる情報伝播経路を構成する．MHA の情報処理は先に述べた点群変換の観点から見ると，トークン間の文脈情報を利用して動的に配置を調整した点群を複数生成する機構として機能している．

3.3 Multi-Layer Perceptron

MHA 層の出力は，複数の文脈依存の特徴表現を生成するが，線形変換を用いた Query，Key，Value の構成では，その特徴の表現力は限定的なものとなる．これを克服するために，MHA 層から得られたヘッドごとに異なる表現ベクトルを各トークン毎に統合し，Multi-Layer Perceptron (MLP) による非線形変換を用いて特徴の表現力を向上させている．

MLP は Single-Layer Perceptron (SLP) の合成として定義される．SLP は $\mathbb{R}^p$ から $\mathbb{R}^q$ への写像で，

$$\mathrm{SLP}(\boldsymbol{x}) = \phi(W\boldsymbol{x} + \boldsymbol{b}), \quad \boldsymbol{x} \in \mathbb{R}^p,\ \boldsymbol{b} \in \mathbb{R}^q,\ W \in \mathbb{R}^{p\times q}$$

で定義される．ここで ϕ は活性化関数と呼ばれる成分ごとに作用する関数で，例えば

$$\phi(z) = z \qquad \text{(線形; linear)}$$
$$\phi(z) = \max\{0, z\} \qquad \text{(Rectified Linear Unit; ReLU)}$$

などが用いられる．Transformer では 1 層目に ReLU，2 層目に線形の活性化関数を持つ 2 層の MLP を用いることが多い:

$$\mathrm{MLP}(\boldsymbol{x}) = \mathrm{SLP}_2 \circ \mathrm{SLP}_1(\boldsymbol{x}) = W_2\ \mathrm{ReLU}(W_1\ \boldsymbol{x} + \boldsymbol{b}_1) + \boldsymbol{b}_2.$$

MHA 層からの出力はトークンごとに MLP に入力し処理される:

$$\boldsymbol{u}_i = \mathrm{CAT}\left(\{\boldsymbol{u}_i^{(h)}, h \in \mathcal{H}\}\right), \quad \text{(複数ベクトルを連結)}$$
$$\boldsymbol{x}'_i = \mathrm{MLP}\left(\mathrm{CAT}(\{\boldsymbol{u}_i^{(h)}, h \in \mathcal{H}\})\right) = \mathrm{MLP}\left(u_i\right), \quad i \in \mathcal{I}.$$

Attention 機構の基本構成要素である MHA と MLP は極めて単純な構造であるが，相互に連携して複雑な特徴表現を生成し，Transformer モデルの強力な表現能力を支えている．特に，MHA は文脈依存の情報を複数の視点から抽出し，MLP は非線形性を導入することでモデルの表現力を大幅に向上させていると考えられる．なお，MHA と MLP の出力がそのまま次の層へ渡されるわけではなく，多くの Transformer 実装では残差結合 (Residual Connection) と層正規化 (Layer Normalization) が標準的に組み込まれている．これらは深層ネットワークの勾配消失問題を緩和し，層を深く積み上げた際の学習安定性を保証する重要な機構であるが，ここでは詳細には踏み込まない．

以上で整理した MHA と MLP を総合すると，Transformer の符号器は入力点群を層ごとに段階的に編集する動的な情報変換器であることがわかる．後に見るように，入力に近い下層では局所的な文脈統合 (近傍トークンの結合) が優勢であり，中層では意味的・構文的な因子の分離・統合が進行し，上層に近づくにつれて文全体を横断する大域的な情報統合へと移行するが，これらの構造的特性を踏まえ，その役割分担を視覚化するために，次節では Integrated Gradients と Layer-wise Relevance Propagation を組み合わせた解釈手法を説明する．MHA のグラフ重みと MLP の非線形写像がどのように連携し，出力の生成根拠を形成していくのかを，情報伝播経路の可視化を通じて明らかにする．

4 情報処理過程の解釈

Transformer は単純な構造の MHA と MLP の組み合わせで構成されているものの，内部でどのような情報処理が行われているのかを逐一追跡し把握することは難しい．内部の情報処理を簡略化した形で数量化し，視覚化する試みは多数行われている．

簡便でよく用いられる方法は Attention Weight の行列表示である Attention Map [1] を利用するものである．Attention Weight はトークン i の表現を更新する際にトークン j にどれだけ注目するかを表す重みであるため，モデルの内部の計算過程におけるトークン間の関係を可視化することになる．これにより，モデルが何に注目して文脈を判断しているかを検証することができる．また，翻訳や要約などにおいてモデルが誤った推測を行った場合の原因を特定する手がかりとなり，モデルの改善に向けたフィードバックを提供することができる．

Attention Map の利点は明確であるが，その問題点を指摘している研究もある．例えば Jain ら [8] は Attention Weight と入力の感度解析 (勾配や交叉検証) の結果が低相関であること，Attention Weight

をランダムに置換しても性能がほとんど変わらないケースが多いことを実験的に示し，Attention Map のみを用いてモデルの出力を説明すべきではないと主張している．Attention Map は Transformer の内部挙動を可視化する代表的な手法であるが，Attention Weight のみを用いるため，実際に各トークンからどのような情報が伝播しているかまでは十分に捉えられない．この問題に対し，Kobayashi ら [9] は Attention Weight に Value ベクトルを組み合わせることで，Attention による情報伝播をより実態に即して分析する手法を提案した．

しかし，このような手法は主に Attention 層に着目しており，Transformer のもう一つの重要な構成要素である MLP 層の影響を十分に考慮できない．そこで近年では MHA と MLP の双方を対象に貢献度を解析する AttnLRP [10] や，モデル内部表現を Sparse Autoencoder (SAE) などの解釈可能な特徴空間に写像し，影響度の高い経路を追跡する Circuit Tracing [11] などの手法が提案されている．

一方で，これらの手法にも課題が残る．例えば AttnLRP はモデル全体に対する貢献度伝播を扱える一方で，計算コストが高くなる場合がある．また，貢献度を各構成要素へ分配する過程で近似誤差が生じる可能性もある．さらに Circuit Tracing は，内部計算を特徴間の経路として可視化できる強力な手法であるが，SAE などによってモデルの表現を置き換えて解析するため，元のモデル内部の計算をそのまま直接観察しているわけではないという制約がある．

以下では，Transformer の情報処理過程をより詳しくみるために，Fig.1 に示したグラフ上で，Integrated Gradients (IG) [12] と Layer-wise Relevance Propagation (LRP) [13] を用いて情報の流れを可視化する方法を紹介する．

4.1 Integrated Gradients

Integrated Gradients (IG) は，多入力 1 出力の可微分な非線形関数の出力値に対して，入力の各成分がどれだけ貢献したかを測る方法の一つで，非線形関数を入力点付近で局所的に線形化して解釈性を付与する．以下では k 次元入力の関数 f の入力 $\boldsymbol{x} \in \mathbb{R}^k$ における各成分の寄与を考える．適当な点 $\boldsymbol{x}'$ を取り，$\boldsymbol{x}'$ と $\boldsymbol{x}$ を結ぶ曲線 $c(t)$

$$c : [0,1] \to \mathbb{R}^k, \quad c(0) = \boldsymbol{x}', \quad c(1) = \boldsymbol{x}$$

を考える．このとき

$$f(\boldsymbol{x}) - f(\boldsymbol{x}') = \int_c \nabla f(\boldsymbol{x}) \cdot d\boldsymbol{x} = \sum_{i=1}^{k} \int_c \frac{\partial f(\boldsymbol{x})}{\partial x_i} dx_i = \sum_{i=1}^{k} \int_0^1 \frac{\partial f(c(t))}{\partial x_i} \frac{c(t)}{dt} dt$$

である．曲線 $c(t)$ は始点 $\boldsymbol{x}'$ を含め適切に設計する必要があるが，IG では $\boldsymbol{x}'$ と $\boldsymbol{x}$ を結ぶ直線

$$c(t) = \boldsymbol{x}' + t(\boldsymbol{x} - \boldsymbol{x}')$$

を考え，入力の成分 x_i の寄与 $\tilde{f}_i$ を

$$\tilde{f}_i = f(\boldsymbol{x}') + (x_i - x_i') \int_0^1 \frac{\partial f(\boldsymbol{x}' + t(\boldsymbol{x} - \boldsymbol{x}'))}{\partial x_i} dt$$

で与える．

実装上では，基準点 $\boldsymbol{x}'$ として零ベクトルやランダムな初期埋め込みを用いるのが一般的である．Transformer のベクトル値入力・出力を扱う場合，出力ベクトルの変化量のノルムや，あるいは出力空間において注目する方向 (例えば特定の単語の埋め込みベクトル) との内積をスカラー関数 f として定義し，各入力トークンの勾配積分を計算する．これにより，Attention Weight による単なる注目度ではなく，出力の生成に対して実際に感度を持つ入力成分を定量的に同定できる．下層では局所的な文脈依存，上層では大域的な意味統合に対応する勾配の集中が観測されるため，層間での情報処理の役割分担を数値的に追跡する基盤となる．

4.2 Layer-wise Relevance Propagation

Layer-wise Relevance Propagation (LRP) は，非巡回グラフ，特にニューラルネットワークのような層状の二部グラフでの情報の流れを数量化し，出力ノードに対する入力ノードの関連度 (Relevance) を評価するために提案された手法である．簡単のためにまず I 個のノードを持つ出力層と J 個のノードを持つ入力層の二部グラフを考える．出力ノード i に対して入力ノード j の関連度を r_{ij} で表す．関連度は非負とし，全ての入力ノードからの合計は 1 に正規化されているとする:

$$r_{ij} \geq 0, \quad \sum_{j \in \mathcal{J}} r_{ij} = 1,\ i \in \mathcal{I} = \{1, \ldots, I\},\ \mathcal{J} = \{1, \ldots, J\}.$$

このような二部グラフを層状に重ねることを考える．ノード I 個の出力層，ノード J 個の中間層，ノード K 個の入力層を考え，出力層と中間層，および中間層と入力層でそれぞれ二部グラフを考え，それぞれに非負値の関連度が定義されているとする:

$$\sum_{j \in \mathcal{J}} r_{ij} = 1, \quad \sum_{k \in \mathcal{K}} r_{jk} = 1, \quad \mathcal{I} = \{1, \ldots, I\},\ \mathcal{J} = \{1, \ldots, J\},\ \mathcal{K} = \{1, \ldots, K\}.$$

このとき

$$1 = \sum_{j \in \mathcal{J}} r_{ij} \left(\sum_{k \in \mathcal{K}} r_{jk} \right) = \sum_{k \in \mathcal{J}} \left(\sum_{j \in \mathcal{J}} r_{ij} r_{jk} \right)$$

が成り立つので，出力ノード i に対する入力ノード k の関連度 r_{ik} は

$$r_{ik} = \sum_{j \in \mathcal{J}} r_{ij} r_{jk}$$

となることがわかる．

更に多層となる場合やより複雑な場合も同様にして計算することができる．これにより LRP は Transformer の層状 DAG (Directed Acyclic Graph) 構造を考慮しつつ，出力の根拠を入力側へ逐次逆伝播させることにより，どの経路が出力生成に寄与したかを定量的に評価することができる．

4.3 IG と LRP の統合

上記の IG と LRP は，単独でも有用であるが，本稿では両者を統合した枠組みを用いて Transformer の情報処理を可視化する．IG が提供するのは MHA 層および MLP 層での出力に対する入力の寄与であり，LRP が提供するのは内部計算グラフ上での関連度の保存的伝播である．

MHA 層では，Attention Weight w_{ij} と Value ベクトルによる勾配情報を組み合わせ，出力側のトークン i に対する入力側のトークン j の寄与を計算し，これを関連度 r_{ij} とする．具体的には，Attention 層のあるヘッドから得られるトークン i の表現ベクトルを

$$\boldsymbol{u}_i = \mathrm{ATT}_i(\{\boldsymbol{x}_j, j \in \mathcal{I}\})$$

と書くことにし，全てのトークンが x_i となる状況を基準点として以下のベクトルを考える:

$$\boldsymbol{u}_i(t) = \mathrm{ATT}_i(\{\boldsymbol{x}_i + t(\boldsymbol{x}_j - \boldsymbol{x}_i), j \in \mathcal{I}\}).$$

これを用いてトークン i への寄与を考えるための関数 $f(c(t))$ を

$$f(c(t)) = \|\boldsymbol{u}_i(t) - \boldsymbol{u}_i(0)\|_2$$

によって定義する．ただし関数 ReLU は成分ごとに作用する．

MLP 層では，活性化関数 (ReLU 等) の閾値特性を反映し，正の寄与のみを伝播させるルールを適用して各トークン i ごとにヘッド h の寄与を計算し，これを関連度 r_{ih} とする．MHA 層と同様であるが，基準点を原点として以下のベクトル

$$\boldsymbol{x}_i'(t) = \mathrm{MLP}\left(\mathrm{CAT}(\{t\boldsymbol{u}_i^{(h)}, h \in \mathcal{H}\})\right)$$

を考え，寄与を考えるための関数 $f(c(t))$ を

$$f(c(t)) = \|\boldsymbol{x}_i'(t) - \boldsymbol{x}_i'(0)\|_2$$

によって定義する．

MHA と MLP からなる Transformer の計算グラフに従って，各層での関連度の流れを計算する．これらを組み合わせることで，出力側から入力側に層間を跨いで遡って関連度の高い重要な経路を追跡することが可能となる．

次節では，この統合手法を用いた可視化実例を示す．入力文に対する高関連度経路の抽出結果を層方向・トークン方向・ヘッド方向の多次元プロットで表示し，Transformer が内部でどのように点群を動的に再構成しているかを，直感的に理解できる形での視覚化を試みる．

5 可視化の例

以下では Transformer の実装の一つである BERT [14] に対して LRP と IG を用いての情報処理過程の可視化を行った例を示す．BERT は Transformer 層を 12 層 (入力側が Layer 0，出力側が Layer 11) 持ち，各層の MHA のヘッド数は 12 (Head 0 から 11 と表記) となっている．

5.1 トークン間の寄与

まず MHA と MLP の寄与から各 Transformer 層での関連度がどのように計算されるかを示す．Figure 2 は入力文 (空白でトークンに分割) ”The firm ’s drop in net reflected weaker revenue in transactions for its own account – a decline of 19 % to $ 314.6 million on reduced revenue from trading fixed-income securities . ” に対する各層でのトークン同士の関係を示している．前層 (トークンで表示) から次層 (番号で表示) への寄与を円の大きさで表し，赤が対象トークン自身からの寄与を表す．この例では，Layer 0 などの入力付近の層では寄与が主として当該トークンの周辺に集中し，近隣のトークン (語句) の関係が強いことがわかる．一方 Layer 4 付近からは，” The firm ’s drop in net ” という表現への寄与において，文章内の位置では離れた ” a decline of 19 % ” など，意味的に結びつくトークンの影響度が相対的に高まる．Layer 9 から Layer 11 にかけては，寄与の分布がより一様になっている．したがって，処理の初期では局所的な関係の整理，中間ではある程度離れた位置の意味的あるいは構文的に重要な関係の整理，終盤では文全体を横断する情報の統合という，層に応じた役割分担が読み取れる．

5.2 重要な経路の探索

各層において，MHA と MLP を継いだ入出力関係を考え，これらの接続関係をエッジとみなした有向グラフ上で，あるトークンの出力に至るまでの経路を考える．MHA ではヘッドごとにトークン同士の関係を，MLP ではトークンごとにヘッドとトークンの関係を考えることになる．あるトークンの出力においてどの経路が情報処理として重要かを示すために，グラフ上で関連度（寄与の積）が大きい経路を複数本求める．ここでは Beam 探索で近似的に求めている（Beam size 300, MLP 側 Top-K 5，ATT 側 Top-K 10）．

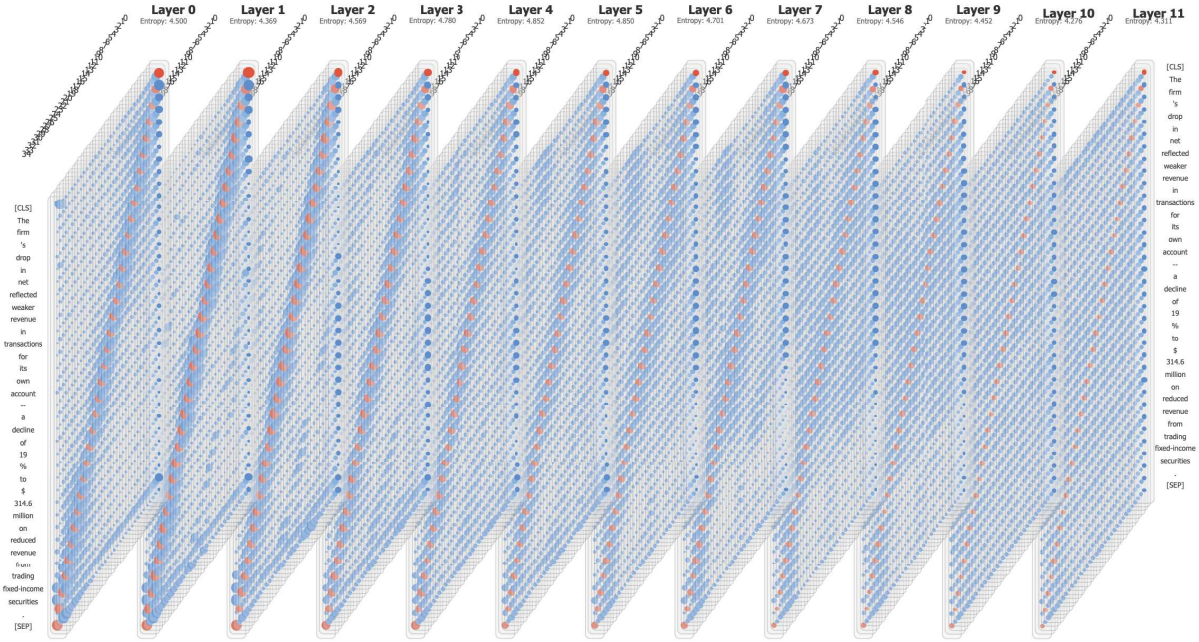

Figure 2: 各層におけるトークン寄与の可視化の例．入力文 “The firm ’s drop in net reflected weaker revenue in transactions for its own account – a decline of 19 % to $ 314.6 million on reduced revenue from trading fixed-income securities . ” に対して層間の寄与を積み上げて表示したもの．

具体例として Figure 3 に示したように，この可視化では最終的なトークン表現が，どの層・どのヘッド・どの入力トークン経由で積み上がるかを，経路に沿った寄与として提示する．その結果，深い層ほどターゲットごとに経路の分岐がはっきりしやすく，入力に近い層では経路が似かよいやすい，という対比が読み取れる．

この傾向は，寄与がネットワーク上で伝播する過程で分散しやすいことと，MHA がトークン間の線形結合（加重平均）として情報を混合することとを踏まえると説明しやすい．すなわち，出力から離れた位置では寄与が薄く広がり，複数ターゲットで共通のハブ的なトークンや層（Figure 3 では Layer 10 付近の 1:The や 4:in）に収束しやすくなる．同時に，層の途中ではターゲット間で分岐の違いも残り，同一入力に対してもトークン固有の重要経路が存在することが示唆される．したがって，各トークンに与えられた最終的なベクトル表現は，単一の共通処理だけでなく，トークンごとに異なる「処理の通り道」への依存を含みうる，と定性的に解釈できる．

6 おわりに

本稿では，Transformer モデルの基本的な構成部品である Multi-Head Attention (MHA) と Multi-Layer Perceptron (MLP) の働きを紹介した．Transformer は単純な構造の MHA と MLP を巧みに組み合わせることによって，複雑な文脈理解と高精度な認識や生成を可能にしている．

Transformer の登場は，単なる言語処理を超えて，エージェント化やマルチモーダル処理への展開を可能にし，人間の思考に近い自律的な言語処理の実現へと向かう新たな道を開いた．Transformer の大規模化とともに Transformer を越える新たな情報処理構造の模索も始まっており，今後はより複雑な文脈や非言語的情報を統合したモデルの開発が期待される．

参考文献

1. Vaswani, A. *et al. Attention Is All You Need* in *Advances in Neural Information Processing Systems* **30** (Curran Associates, Inc., 2017).
2. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. & Dean, J. *Distributed Representations of Words and Phrases and Their Compositionality* in *Advances in Neural Information Processing Systems* **26** (Curran Associates, Inc., 2013).
3. Le, Q. & Mikolov, T. *Distributed Representations of Sentences and Documents* in *Proceedings of the 31st International Conference on Machine Learning* International Conference on Machine Learning (PMLR, June 18, 2014), 1188–1196.
4. Hochreiter, S. & Schmidhuber, J. Long Short-Term Memory. *Neural Computation* **9,** 1735–1780 (Nov. 15, 1997).
5. Adadi, A. & Berrada, M. Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access* **6,** 52138–52160 (2018).
6. Barredo Arrieta, A. *et al.* Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI. *Information Fusion* **58,** 82–115 (June 1, 2020).
7. Molnar, C. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable* 3rd ed. (2025).
8. Jain, S. & Wallace, B. C. *Attention Is Not Explanation* 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)* NAACL-HLT 2019 (eds Burstein, J., Doran, C. & Solorio, T.) (Association for Computational Linguistics, Minneapolis, Minnesota, June 2019), 3543–3556.
9. Kobayashi, G., Kuribayashi, T., Yokoi, S. & Inui, K. *Attention Is Not Only a Weight: Analyzing Transformers with Vector Norms* in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* EMNLP 2020 (eds Webber, B., Cohn, T., He, Y. & Liu, Y.) (Association for Computational Linguistics, Online, Jan. 1, 2020), 7057–7075.
10. Achtibat, R. *et al. AttnLRP: Attention-Aware Layer-Wise Relevance Propagation for Transformers* in *Proceedings of the 41st International Conference on Machine Learning* International Conference on Machine Learning (PMLR, July 8, 2024), 135–168.
11. Ameisen, A. E. *et al. Circuit Tracing: Revealing Computational Graphs in Language Models* Transformer Circuits. https://transformer-circuits.pub/2025/attribution-graphs/methods.html (2026).
12. Sundararajan, M., Taly, A. & Yan, Q. *Axiomatic Attribution for Deep Networks* in *Proceedings of the 34th International Conference on Machine Learning* International Conference on Machine Learning (PMLR, July 17, 2017), 3319–3328.
13. Montavon, G., Binder, A., Lapuschkin, S., Samek, W. & Müller, K.-R. in *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning* (eds Samek, W., Montavon, G., Vedaldi, A., Hansen, L. K. & Müller, K.-R.) 193–209 (Springer International Publishing, Cham, 2019).
14. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. *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)* NAACL-HLT 2019 (eds Burstein, J., Doran, C. & Solorio, T.) (Association for Computational Linguistics, Minneapolis, Minnesota, June 2019), 4171–4186.

Faculty of Sciences and Engineering
Waseda University
Tokyo 169-8555, JAPAN
E-mail address: noboru.murata@eb.waseda.ac.jp

早稲田大学・理工学術院 村田 昇

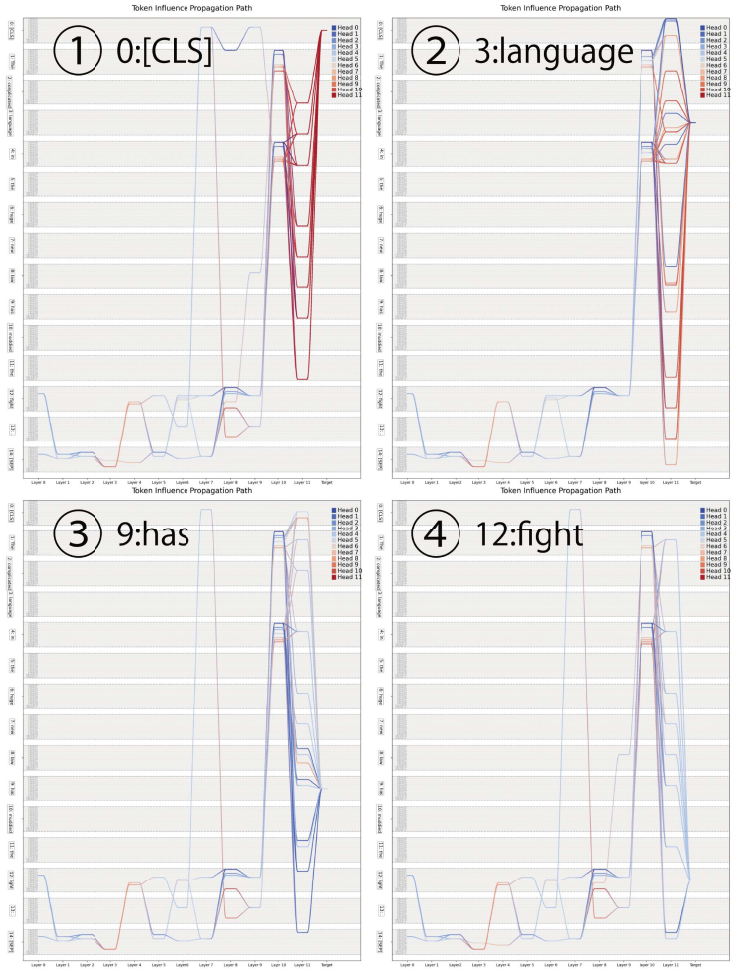

Figure 3: 重要な経路の可視化．入力文 ”The complicated language in the huge new law has muddied the fight .” に対し，ターゲットを 0:[CLS]，3:language，9:has，12:fight とした場合の重要経路を示す（Beam size 300，MLP 側 Top-K 5，ATT 側 Top-K 10）．横軸は層（左が Layer 0，右が Layer 11），縦軸は入力トークン位置に沿った Head 0 から Head 11 の配置である．

① （0:[CLS]）では Layer 11 の MLP において Head 11 の寄与が強く，ATT では Head 11 経由で $u_{[CLS]}^{(11,11)}$ に至る経路が 2:Token，3:language，4:in，6:huge，7:new，8:law，9:has，11:the などへ分岐し，文全体を見渡すようなパスとなる．0:[CLS] 起点の経路は相対的に弱く，最終層付近で情報を束ねる役割と読める．

② （3:language）では Layer 11 の MLP が Head 0，1，8，9，10 などへ分散し特定 Head への一極集中は見られず，ATT では [CLS] も参照するため Layer 11 内だけでも複数経路が並立する．自身の language に加え，8:law や 12:fight 側へのつながりも確認できる．

③ （9:has）では Layer 11 の MLP に Head 1，4，8 などへの寄与が集まり，目的語 12:fight への係りが視覚的に確認できる．

④ （12:fight）では Layer 11 の MLP で Head 1，4 の寄与が目立つ．Layer 10 付近ではいずれのパネルも 1:The と 4:in を通る部分が共通しており，LRP における寄与の分散と，MHA による加重平均としての意味混合と整合的である．一方 Layer 7 付近では ① と ④ で 0:[CLS] の Head 4 に関わる経路が見えるのに対し，③ は同じ位置を主経路としてほとんど通らず，① の Layer 7 における 0:[CLS] の Head 4 への至り方は ③ や ④ と異なる．さらに Layer 9 の [CLS] の Head 4 や Layer 8 の 1:The の Head 1 を通る経路など，ターゲット間の差異も残る．

以上より，トークンごとに重要経路の通り方が異なり，トークン固有の経路がその表現形成において影響の大きい処理部位を示唆するものと解釈できる．