GPT-2 からの bigram 知識の取り出し
Bigram Knowledge Extraction from GPT-2

吉田 稔 *　　松本 和幸 †
Minoru Yoshida　　Kazuyuki Matsumoto

概要

GPT-2 から bigram 知識を取り出す手法を提案する．提案手法は，第 1 層のヘッドに着目し，出力単語を予測する．さらに，誤差逆伝搬により，与えられた bigram から，それと関連する文脈語を取り出す手法も提案する．実験では，提案手法がベースライン手法と比べ精度の高い抽出を行えることを確認する．
キーワード：GPT
Keywords: GPT

1　はじめに

本稿[*1]では，GPT 言語モデルから，**単語 bigram 知識を取り出す**手法を提案する．大規模言語モデル (Large Language Models, LLM) は，近年の自然言語処理で標準的に使われるリソースである．大規模なコーパスを対象に，教師データを自動的に作成する**自己教師あり学習**を行い，多層ニューラルネットワーク等の機械学習モデルに，一般的な言語知識を，パラメータとして獲得させる．大規模言語モデルのパラメータを利用して必要な個別タスクを解かせることにより，事前学習なしで個別タスクに対する学習を行った場合よりも，高い性能が得られることが知られている．[Devlin 19]

GPT(Generative Pretrained Transformers) 言語モデル [Radford 18, Radford 19, Brown 20] は，Transformer デコーダ [Vaswani 17] に対し，自己教師あり学習タスクとして，「与えられた単語列から，次の単語を予測する」というタスクを与えた大規模言語モデルである．GPT を利用することにより，与えられた入力文に続く自然な文を生成できる．さらに，入力文（プロンプト）を工夫することにより，機械翻訳，要約等の様々なタスクを，少量の学習データを用いるか，あるいは学習データを全く用いずに解かせることができることが報告されている．[Brown 20]

しかしながら，GPT をはじめとする大規模言語モデルが，内部でどのように言語知識を学習しているのか，様々な分析が行われているものの，その原理については，まだ不明な点が多い．

本研究では，LLM と言語知識を結びつける新たなアプローチとして，**GPT-2 のパラメータから bigram 知識を取り出す**手法を提案する．一般的に，大規模言語モデルのパラメータの保存に必要な容量は，その学習に用いられた大規模なコーパスの保存に必要な容量に比べ，遥かに小さい．すなわち，コーパスの性質に関する知識が，大規模言語モデルのパラメータとして圧縮して保存されていると考えることができる．ここからの

* 責任著者，徳島大学理工学部 (Faculty of Science and Technology, Tokushima University)，mino@is.tokushima-u.ac.jp
† 徳島大学理工学部 (Faculty of Science and Technology, Tokushima University)
*1 本原稿は，査読前のプレプリントです．

言語知識の取り出し手法を開発することにより，より効率的な言語知識の蓄積・取り出しを実現できることが期待される．また，GPT-2 は，次単語の予測を行うモデルであり，その知識は bigram 知識を包含していると考えられるが，実際にモデルのどの部分に知識が蓄積されているかは明らかではない．GPT-2 から bigram 知識を抽出する手法を研究することにより，GPT-2 モデルのどの部分に bigram の知識が蓄えられ，それが次単語予測にどのように応用されているかのしくみの理解に役立つと考えられる．bigram 知識は，テキストからの知識獲得の中でも，最も基本的かつ重要な知識であり，人間にも理解が容易である．LLM が持つ bigram 知識を調査することで，例えば，複数の異なる LLM についての性質を比較する，あるいは，学習データが明らかになっていない LLM について，モデルの学習に用いられたコーパスについての性質を理解する等の応用が考えられる．

2 関連研究

近年の LLM の発展に伴い，その挙動や仕組みを明らかにしようとする試みが増えてきている．本節では，LLM の分析に関する既存研究について述べる．

2.1 パラメータに直接着目した手法

Transformer のパラメータを直接分析し，どのような言語知識と関連しているかを分析する手法が，多くの先行研究において用いられている．典型的なアプローチとして，言語知識を検証するためのタスク（出力単語の時制判別など）を，モデル内部の各パラメータ（出力の Embedding 等）を説明変数として利用し解かせる Probing という手法 [Lin 19, Koto 21] や，Attention の重みを分析するものがある [Clark 19, Lin 19, Marecek 19, Reif 19, Vig 19, Voita 19]．Attention 重みは，各 Head において入力単語どうしの関連の強さを分析するのに役立つ．

Oba ら [Oba 21] は，各ニューロンを活性化させる文の集合をマイニングすることで，ニューロンと言語知識を対応付ける手法を提案している．また，Bricken ら [Bricken 23] は，MLP 層の隠れ層におけるニューロンから多義性を除去して意味のある特徴量を抽出するため，辞書学習とスパース学習を利用した手法を提案している．

よりパラメータ間の役割の差異を明らかにするアプローチとして，説明対象のモデルの一部分のみを使用し，残りの部分を枝刈り（pruning あるいは knockout）するというアプローチをとる研究もある．目的関数を最大化させるような枝刈りを求め，残った部分が対象の言語現象に関わる部分構造ということになる．Michel ら [Michel 19] は，Transformer による機械翻訳や推論（NLI）タスクを対象に，精度に貢献する Head がわずかであり，その他の Head を枝刈りしても精度を大きく下げないことを報告している．Voita ら [Voita 19] も同様に，Transformer による翻訳タスクを対象とし，精度に貢献する Head を発見する手法を提案している．さらに，精度に貢献しない Head を，精度をあまり下げずに枝刈り（pruning）できることを示している．Held ら [Held 23] は，多言語 Transformer における interference という現象に対し，ゲーム理論における Shapley 値という概念を用い，Shapley head pruning という手法を提案している．

本研究で用いる手法も，特定の Head のみを用いて（残りの Head の重みを 0 とすることにより）後続単語を導出するという点は，Pruning と同一であるが，入力単語に依らず第 1 層の Head のみを用いるという手法は，本研究独自のものである．

2.2 Transformer モデルの全体像に関する研究

Transformer のモデル全体を対象とした分析も進んでいる．Ferrrando ら [Ferrando 22] は，Attention が出力に与える影響を，Attention 重みを Head 内だけで扱うのではなく，モデル全体の情報の流れを考慮することで，出力に対する各 Attention の重要度の和をとり，重要語を抽出する手法を提案している．

Attention 重み以外に着目した研究としては，FFN に関する分析 [Geva 21, Geva 22] や，各層を接続する残差接続および正規化層についての分析 [Kobayashi 21] も報告されている．特に，Geva らの研究 [Geva 21, Geva 22] では，GPT 等の自己回帰言語モデルにおいて，出力単語（次単語）の知識が FFN 層に分散して蓄積されていることを報告しており，これは，本研究の主張と合致するものである．本研究では，FFN 層だけでなく，第 1 層の MHA 層にも同様に出力単語（次単語）の知識が蓄積されていることを主張し，その検証を行う．また，Value-Output ベクトルの分析を行った既存研究として，Gupta らは，Value 変換を考慮した場合と考慮しない場合を比べ，Value 変換を考慮しない場合に，Attention 重みと出力に齟齬が生じることを報告している [Gupta 21]．Kobayashi らは，入力ベクトルがどのようにして Value-Output ベクトルに変換されるかに着目し，変換後のベクトルのノルムを分析することで，ノルムと単語頻度の関連等を分析している [Kobayashi 20]．また，吉田ら [吉田 23] は，Value-Output 変換のパラメータ行列（以下，Value-Output 行列）の性質そのものに着目した分析を提案している．

2.3 単語空間への射影を利用した手法

パラメータを単語 Embedding の空間に射影することにより，パラメータの意味を単語を利用して理解しようとする試みも多く提案されている．

Logit lens [nostalgebraist 20] は，GPT の各 Layer の出力を Embedding Space に射影することで，出力が Layer 毎にどのように変化するかを観察する手法である．これらをより洗練させた手法として，Langedijk ら [Langedijk 23] らは，DecoderLens を，Belrose ら [Belrose 23] は Tuned Lens を提案している．これらの手法でも，GPT の出力が，Layer の浅い段階から，入力の Embedding 空間から，出力（後続語）の Embedding 空間に切り替わることが報告されている．本研究では，これをより具体的に，第 1 層の Value ベクトルに多くの bigram 知識が蓄積されていることを確認する．さらに，Sakarvadia ら [Sakarvadia 23] は，同様の考え方を各 Head の出力に適用した Attention Lens という手法を提案している．

Dar らは，Transformer の各 head の役割を，Embedding への射影で分析するアプローチを提案している．Value-Output 変換のパラメータ行列を単語分散表現と関連づけ，関連が深い単語ペアを取得する手法を提案している [Dar 23].

本研究も，モデルパラメータと単語を関連づけるという意味では同様のアプローチであるが，本研究では，各層の出力を直接用いるのではなく，他の層を経由したモデル全体の出力を射影した結果を用いる．

Baeumel ら [Baeumel 23] は，BERT の各ニューロンに関係する単語を発見するため，どのような入力単語が与えられた場合に対象ニューロンが活性化するかを，Gradient Ascent を用いて発見する手法を提案している．これは，本研究で提案する文脈語取得手法と同一の考え方である．

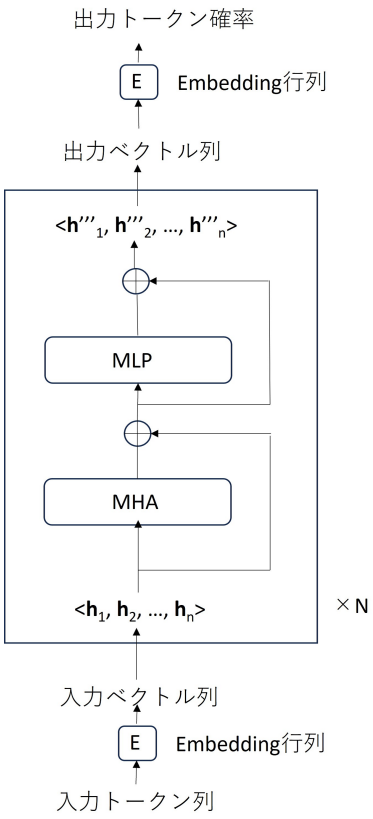

図 1　GPT-2 の概要（Layer Normalization は省略）

2.4　GPT の挙動の説明

特定のパラメータや言語現象に限定せず，GPT 言語モデル全体の挙動について，統一的に分析するアプローチも提案されている．

Gurnee ら [Gurnee 23] は，GPT の変種である Pythia というモデルを対象とし，probing による言語現象の説明の際に，使用するニューロンの上限を定めることで，言語現象に関連するニューロンを発見する手法を提案している．複合語に関するニューロンについても実験しており，これは，後続語とモデル内部構造の関係についての議論とみることもできる．

Bills ら [Bills 23] は，GPT-4 に GPT-2 の挙動を説明させる，というアプローチを報告している．この中で，特に，高次のレイヤーでは，現在の単語ではなく，次の単語の予測に関わる挙動が見られる，という報告がされている．本研究では，低次のレイヤーも後続単語予測に重要であることを提案手法を通じて示す．

3　GPT-2

ここでは，本研究の分析対象とする GPT-2[Radford 19] の全体像について説明し，その後，その一部である Multi-Head Attention および Multi-Layer Perceptron について詳しく説明する．なお，本稿では，語彙

知識の取り出しが目的のため，position embedding については考慮しない．また，Dropout については無視する．これは，一般に，Dropout が学習時にのみ用いられるのに対し，本研究では，推論時の GPT-2 の挙動について分析しているためである．

GPT-2 は，**言語モデル**と呼ばれる．これは，入力として単語の列が与えられたとき，それに後続する単語を予測するモデルである．例えば，`<Mt., Fuji, is, the, Japanese, highest>`という入力から，後続する単語を `mountain` であると予測する．これは，GPT-2 においては，「各単語に対応する入力ベクトルが，Transformer 変換により，後続単語に対応するベクトルとなる」という挙動で実現される．この例では，入力の各単語が出力において`<Fuji, is, the, Japanese, highest, mountain>`と変換されれば正解となり，従って，後続単語の予測は，入力の最終単語（この場合は `highest`）がどのような出力ベクトルに変換されるかによって決まる．この場合，最終単語の入力ベクトルが，出力ベクトルにおいて，次の単語 (`mountain`) に近いベクトルに変換されれば正解となる．

1 に，GPT-2 の概略図を示す．GPT-2 は，Transformer エンコーダ [Vaswani 17] に基づくモデルである．Transformer は，入力として与えられた単語分散表現ベクトルの系列を，複数の層を通じて変換していくモデルであり，各層には，Multi-Head Attention (MHA) 層と，Multi-Layer Perceptron (MLP) 層[*2]の各ネットワークが含まれている．このうち MHA 層では，同一文内の別の単語（GPT の場合は，現在の単語より前の単語）からの影響を計算している．また，MLP 層には，言語一般の連想知識が蓄積されているとの報告がある [Geva 21]．入力単語列は，各単語に対応する埋め込み表現 (単語分散表現) を定義した **Embedding 行列**によりベクトルの列に変換され，L 層の Transformer ブロックに入力される．入力ベクトルは，Transformer の各層を通じて段階的に変化し，最終層の出力ベクトルとなる．結果として，1 つの入力ベクトルに対し，1 つのベクトルが出力される．最終層の出力ベクトルと，出力用の Embedding 行列[*3]との積をとり，Softmax 関数を通すことで，出力単語の確率を計算する．各層は残差接続されているため，入力ベクトルが段階的に出力ベクトルに変化すると見なすことができる．

以上を定式化すると，以下のようになる．第 l 層への入力を，ベクトルの列 $H =< \boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_n >$ とする．[*4] 各 GPT-2 ブロックは，まず，入力に対し，以下の変換を行う．

$$< \boldsymbol{h}'_1, ..., \boldsymbol{h}'_n >= \mathrm{MHA}_i(\mathrm{LN}_{i1}(< \boldsymbol{h}_1, ..., \boldsymbol{h}_n >)) \tag{1}$$

その後，各入力 $\boldsymbol{h}_j$ に対し，

$$\boldsymbol{h}''_j = \boldsymbol{h}'_j + \boldsymbol{h}_j \tag{2}$$

$$\boldsymbol{h}'''_j = \mathrm{MLP}_i(\mathrm{LN}_{i2}(\boldsymbol{h}''_j)) + \boldsymbol{h}''_j \tag{3}$$

と変換する．この $\boldsymbol{h}'''_j$ がそのまま次の層の j 番目の入力となる．ここで MHA(x) は，MHA 層（後述）による変換を，MLP(x) は，MLP 層による変換を表す．式 2 と 3 の第 2 項は残差接続を表す．また，LN(x) は Layer Normalization を表す．[*5]

最終層の出力 $\boldsymbol{h}'''^{L}_j$ には，以下の変換が行われ，最終的に softmax 関数により出力単語の確率が与えられる．

$$\mathrm{softmax}(W_e \mathrm{LN}_{final}(\boldsymbol{h}'''^{L}_j)) \tag{4}$$

[*2] Feed-Forward Network (FFN) 層とも呼ぶ

[*3] 多くの実装では，入力の Embedding 行列と同じものが用いられる．

[*4] ここで，n は文の長さ（入力単語数）である．簡単のため，以下の式では，特に必要ない場合，Layer 番号 l は省略する．

[*5] 各 Layer Normalization は，固有のパラメータを持つため，i でインデックス付けしてある．

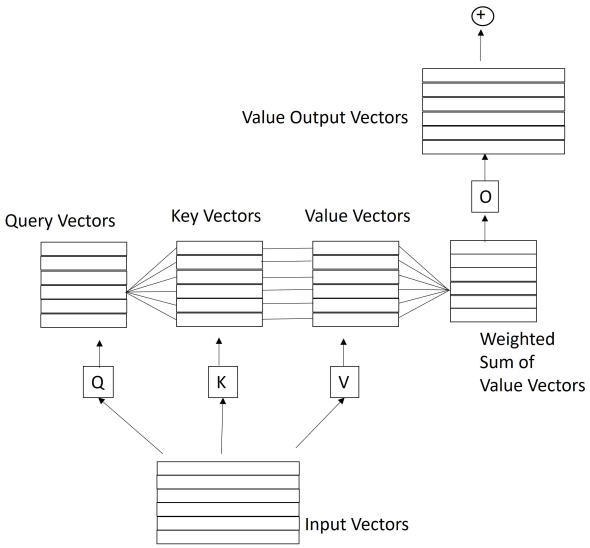

図 2　Multi Head Attention の概要

なお，出力単語の選択の際は，一般的には，入力 Embedding と同一の行列が用いられるため，本稿でもその設定を採用している．

3.1　Multi-Head Attention

Multi-Head Attention(MHA) は，複数の Head と呼ばれるサブモジュールから構成される．各 MHA モジュールにはいくつかの Head が並列して存在しており，各 Head には同一の入力ベクトル列が与えれ，別々の出力ベクトルを返す．それぞれの Head の出力ベクトルの和をとり，最終的な MHA の出力となる．各 Head には，単語間の関連性についての様々な知識が，行列のパラメータとして保存されていると考えられる．ここで，Kobayashi らの研究 [Kobayashi 20, Kobayashi 21] と同様，各 Head について，入力 $\boldsymbol{x} \in \mathbb{R}^d$ を変換する関数として以下に説明する．[*6]

以下，MHA への入力を，ベクトルの列 $< \boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n >$ とする．この入力は，そのまま各 Head への入力となる．各 Head のパラメータを

$$W_Q, W_K, W_V \in \mathbb{R}^{d \times d'} \qquad \boldsymbol{b}_Q, \boldsymbol{b}_K, \boldsymbol{b}_V \in \mathbb{R}^{d'}$$

で表現する．ここで，d は入力ベクトルの次元数，d' は変換後のベクトル（Head の内部で処理されるベクトル）の次元数である．

2 に，各 Head の入出力の概要を示す．各入力ベクトル列は Q,K,V という 3 種の異なる行列によって線形変換され，それぞれ Query ベクトル，Key ベクトル，Value ベクトルの列となる．

$$\begin{aligned} \boldsymbol{q}_j &= \boldsymbol{x}_j W_Q + \boldsymbol{b}_Q \\ \boldsymbol{k}_j &= \boldsymbol{x}_j W_K + \boldsymbol{b}_K \\ \boldsymbol{v}_j &= \boldsymbol{x}_j W_V + \boldsymbol{b}_V \end{aligned} \tag{5}$$

[*6] 層全体のパラメータ行列を，対応する列に分割することで，各 Head に対応する行列を得ることができる [Dar 23].

ここで，各入力ベクトル $\boldsymbol{x}_j$ に対して，他の入力ベクトルとの関連が，Query ベクトル，Key ベクトルを通じて計算される．具体的には，

- 入力ベクトル $\boldsymbol{x}_j$ に対する Query ベクトル $\boldsymbol{q}_j$
- 各ベクトル $\boldsymbol{x}_k$ $(k \leq j)$ の Key ベクトル $\boldsymbol{k}_k$

との内積

$$\alpha'_{jk} = \boldsymbol{q_j} \boldsymbol{k_k}^\top$$

を計算し，これを softmax 関数により正規化し，単語間の Attention α_{jk} とする．各 Attention α_{jk} は，k 番目の入力ベクトルと j 番目の入力ベクトルが，その Head においてどれだけ関連深いかを，各入力ベクトルへの重みとして表現する．

各単語 j について，α_{jk} で重みづけされた，$\boldsymbol{v}_k$ の重み付き平均を計算する．

$$\boldsymbol{v'_j} = \sum_{k=1}^{n} \alpha_{jk} \boldsymbol{v}_k \tag{6}$$

その後，さらに，同一 Layer の全 Head について和をとった後，全結合層を通し，MHA の出力ベクトル $\boldsymbol{h}'_j$ となる．

この全結合層のパラメータは，各 Layer 毎に，

$$W_O \in \mathbb{R}^{d \times d} \qquad \boldsymbol{b}_O \in \mathbb{R}^d$$

という行列とベクトルで表現される．このとき，MHA の各出力 $\boldsymbol{h}'_j$ は，以下で表現される．

$$\boldsymbol{h}'_j = \sum_h \boldsymbol{v}_j^{'h} W_O + \boldsymbol{b}_O \tag{7}$$

なお，$\boldsymbol{v}_j^{'h}$ は，Head h における v'_j の値を示し，$\sum_h$ は，現在の Layer のすべての Head についての和を示す．

3.2 Multi-Layer Perceptron

Multi-Layer Perceptron (MLP) 層は，MHA 層と違い，ベクトルの列 $< \boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n >$ において，入力ベクトルが他のベクトルの出力に影響を与えることはないため，以下，MLP への入力を，単にベクトル $\boldsymbol{x}$ とする．

MLP 層は，入力を，入力次元よりも大きい次元を持つ空間へ射影する層（Full Connection 層）と，その後ふたたび元の次元へ射影する層（Projection 層）という，2 層の変換を伴う層である．

$$\boldsymbol{m} = f(\boldsymbol{x} W_{M1} + \boldsymbol{b}_{M1})$$
$$\boldsymbol{m}' = \boldsymbol{m} W_{M2} + \boldsymbol{b}_{M2}$$

ここで，f は活性化関数であり，ベクトルの各要素に作用する．

Geva ら [Geva 21, Geva 22] は，MLP 層には，単語の連想知識が入っているとしており，bigram の知識もその一部として含まれていると考えることができる．前述の通り，MLP 層は入力ベクトル同士が影響を与え合わないため，文脈によらない知識の多くがこの層に蓄積されていると考えられる．

4　後続単語集合の取得

前節での説明の通り，GPT-2 では，Attention により同一文の別の単語（文脈単語）の影響を受けて出力が変化する．これは，同一の単語に対しても，それが出現する文の違いによって後続の単語を変化させる働きがある．しかしながら，本研究では，一般的な単語 bigram の知識の獲得を目的としているため，文脈単語の影響を排除することを考える．

ここで，文脈の影響を排除するとは，MHA 層の各 Attention による Value ベクトルの重み付き和（式 (6)）から，自分自身の Value ベクトル $\boldsymbol{v}_j$ の成分のみに着目することを示す．すなわち，式 6 の重みから，α_{jj} のみを残した式

$$\boldsymbol{v}'_{jj} = \alpha_{jj}\boldsymbol{v}_j$$

を考える．

ここで，h 番目の **head の重み** w_h を定義し，$\alpha_{jj} = w_h$ とおく．（$\boldsymbol{v}'_{jj} = w_h\boldsymbol{v}_j$）また，分析のため，$w_h$ の値は，1 か 0 の二値であるとする．すなわち，提案手法は，Head を，採用する Head $w_h = 1$ と採用しない Head $w_h = 0$ に分け，採用する Head の Value ベクトルのみを残す．(従って，採用する Head においては $\boldsymbol{v}'_j = \boldsymbol{v}_j$, 採用しない Head においては，$\boldsymbol{v}'_j = \boldsymbol{0}$ となる．)

このとき，各層の MHA の出力は，式 5, 6, 7 を統合し，

$$\boldsymbol{h}'_j = \sum_{h \in HS(l)} (\boldsymbol{x}_j W_V^h + \boldsymbol{b}_V^h) W_O + \boldsymbol{b}_O \tag{8}$$

となる．ここで，$HS(l)$ は，l 番目の Layer において，採用する Head の集合を表す．また，W_V^h と $\boldsymbol{b}_V^h$ は，それぞれ Head h のパラメータを表す．ここで，各層の入力から出力への変換が式 1, 3 で表されることを考えると，j 番目の入力のみから，（他の入力を用いずに）出力が得られることがわかる．

採用する Head と採用しない Head を分ける理由は，実際には，すべての Head に平均的に bigram の知識が蓄積されているわけではなく，bigram の知識を強く反映した Head と，あまり反映していない Head があると考えられるからである．ここで，実際にどの Head を用いるか（すなわち，どの Head に bigram 知識が蓄積されているか）を考える．

ここで，我々は，様々な Head 重みのパターンを試す中で，「第 1 層の Head の重みを 1 とし，残りの層の Head の重みをすべて 0 とする」ことで，後続単語を得られる傾向があることを発見した．パラメータの学習においては，各層への入力ベクトルと，それ以降の層を通して接続される正解出力ベクトルをもとに，誤差を小さくするように学習が行われる．第 1 層の各入力ベクトルは単語の Embedding ベクトルが直接接続されており，周辺単語の情報が混入していない．このため，出力に用いられる各 Value ベクトルも，Embedding ベクトルに直接 Value 行列を掛けたものとなり，パラメータの学習において，Embedding ベクトルに強く関連した知識が学習されやすいと考えられる．

すなわち，本研究では，bigram 知識が蓄積されるのは，MLP 層および第 1 層の MHA 層であるという仮説を立て，それに基づき bigram 知識を抽出する手法を提案する．

これは，先行研究 [nostalgebraist 20] では，GPT においては，入力単語のベクトルが，早い段階（入力に近い層）で入力単語とはかけ離れた空間に射影され，そこから徐々に出力単語へ変化していくことが示唆されていることと整合性がある．周辺単語ベクトルの影響は，残差接続を通して，層が進むたびに大きくなっていくと考えられるため，低次の層に bigram 知識が蓄積されやすいと考えられるからである．

以下，現在の単語（入力単語）の ID を c とする．前述の通り，GPT-2 は後続単語を予測するモデルであるため，c に後続する単語を予測することになる．

4.1 提案手法: First Layer 法 (FL 法)

上記の通り，我々は，MLP 層および第 1 層の MHA 層に bigram 知識が蓄積されていると仮定する．この仮説に基づき，第 1 層の Head の重みを 1 とし，第 1 層以外の Head の重みを 0 とすることで，MLP 層および第 1 層の MHA 層から bigram 知識を取り出すことを試みる．

このとき，第 1 層においては，式 8 の $HS(l)$ は，第 1 層のすべての Head となる．第 2 層以降の Head については，必ず $w_h = 0$ であるとするため，第 2 層以降においては，式 8 は，

$$\boldsymbol{h}'_j = \boldsymbol{b}_O \tag{9}$$

となり，式 1, 2, 3 の層の入出力変換は，

$$\boldsymbol{h}''_j = \boldsymbol{b}_O + \boldsymbol{h}_j \tag{10}$$

$$\boldsymbol{h}'''_j = \mathrm{MLP}_i(\mathrm{LN}_{i2}(\boldsymbol{h}''_j)) + \boldsymbol{h}''_j \tag{11}$$

となる．すなわち，入力 $\boldsymbol{h}_j$ から出力への変換において，第 1 層においては MHA のパラメータが用いられるが，第 2 層以降では，主に MLP のパラメータのみで変換を行うことになる．

以上の設定において，第 1 層への入力ベクトルを最終層の出力ベクトルに変換したものを，$\boldsymbol{y}_j$ として出力し，$\boldsymbol{y}_j$ と Embedding 行列との積および softmax 関数による変換を経て，出力単語の確率を得る．本稿では，以後，この手法を，First Layer 法 (FL 法) と呼ぶ．

4.2 提案手法 2: FLQK 法

FL 法では，第 1 層のすべての Head の重みを 1 とした．さらに，我々は，第 1 層の重みづけについて，以下で定義する**自己 QK 類似度**を用いることによる，FLQK 法を提案する．

Head h における単語 c の**自己 QK 類似度**を，各単語の類似度で softmax 関数を計算した値

$$\mathrm{selfsim}_h(c) = \frac{\exp(\mathrm{qksim}_h(c,c))}{\sum_j \exp(\mathrm{qksim}_h(c,j))}$$

として定義する．ここで，$\sum_j$ は（c も含む）すべての単語に関する和を示す．また，qksim は，単語 c の query ベクトルと，単語 j の key ベクトルの内積であり，以下で定義される．

$$\mathrm{qksim}_h(c,j) = (\boldsymbol{e}_c W_Q^h + \boldsymbol{b}_Q^h)(\boldsymbol{e}_j W_K^h + \boldsymbol{b}_K^h)^\top$$

ここで，$\boldsymbol{e}_c$ は単語 c の Embedding ベクトル，$\boldsymbol{e}_j$ は単語 j の Embedding ベクトルを表す．

各 Head において，自己 QK 類似度が高い単語は，**周辺に来る単語にかかわらず，自分自身への Self-Attention の値が一番高くなる**ことになるため，実際の Self Attention の計算においても，重みが 1 に近くなると考えることができる．この考察に基づき，各 Head h において，以下の条件を満たすとき，$w_h = 1$ とし，そうでなければ $w_h = 0$ とする．

- $\mathrm{selfsim}_h(c) \geq \alpha_{QK}$

α_{QK} はしきい値であり，現在，$\alpha_{QK} = 0.9$ としている．[*7]

4.3 提案手法 3：MLP 法

上記で説明した FL 法や FLQK 法は，第 1 層の MHA 層を利用することを前提としているが，それとは別に，MHA 層を利用せず，MLP 層の知識だけを利用する手法も考えられる．本論文では，これを，MLP 法と呼ぶ．これは，単純に，すべての MHA 層において重みを 0 とすることで実現される．

5 Bigram からの，文脈単語の取得

前節では，入力単語をもとに，出力単語の候補集合を得る手法 (FL 法[*8]) を提案した．しかしながら，GPT-2 の実際の出力は，これら候補集合の中からランダムに選ばれるわけではなく，文脈に応じた適切な単語が出力される．

この単語選択に関して，我々は，

- 入力 Embedding が第 1 層で連接単語空間（出力空間）に変換され，その後，上位の層の MHA 層において，周囲の単語の情報を取り入れ，文脈に応じた後続単語の選択が行われる

という仮説を立てた．この仮説に基づき，本節では，入力単語 c と出力単語 n が与えられたとき，この出力単語 n（すなわち，bigram $c.n$）を得るために，入力として与えられるべき**文脈単語**を，誤差逆伝搬法により選択する手法を提案する．

FL 法では，実際に MHA において使っている Head は，第 1 層の Head のみである．従って，残りの層の Head を利用することにより，文脈に応じた出力を選択させることが可能となる．この出力の選択に関して，我々は上記の仮説を，以下のように具体化した．

- 第 1 層で得られた後続単語の集合の中から，第 2 層以降に MHA によって混入する文脈語の影響を受けて，正解後続単語 n の方向へ，出力ベクトルが変化する．

FL 法で得られた出力ベクトルを $\boldsymbol{v}_{avg}$ とすると，$\boldsymbol{v}_{avg}$ は，後続単語集合に対応するベクトルの重心ベクトルを表していると考えられる．このとき，文脈単語から混入するベクトルは，$\boldsymbol{v}_{avg}$ を，後続単語の Embedding ベクトル $\boldsymbol{e}_n$ に変化させるようなベクトルであると考えられる．

この流れを逆に辿ると，出力ベクトル $\boldsymbol{v}_{avg}$ を $\boldsymbol{e}_n$ 方向に動かすような文脈単語ベクトルを推定すれば，後続単語 n を導くような文脈単語が得られることが予想される．

ここで，各単語に対する出力ベクトルは，式 4 により，

$$\mathrm{LN}_{final}(\boldsymbol{h}^{'''L}) \tag{12}$$

であることがわかる．この出力ベクトルと，各単語分散表現ベクトルの内積をもとに，出力単語が選択されることになる．

ここで，第 2 層以降の Attention の値が，すべて 1 つの文脈単語 k に割り当てられていると仮定する[*9]．す

[*7] $selfsim$ は softmax 関数により定義されているため，しきい値の変化は精度にあまり影響が無いと考えられる．実際に，$\alpha_{QK} = 0.5$ とした場合を調査したところ，精度にあまり差は出なかった．

[*8] 実験では，FL 法と FLQK 法の両方を用いるが，簡単のため，以下の説明では，FL 法についてのみ言及する．

[*9] ここで，k は，入力文中の index を表す．

なわち，第 2 層以降のすべての Head に，文脈単語 k の成分を混入させることを考える．

このとき，式 9 は，

$$\boldsymbol{h}_j^{'l} = \sum_h (\boldsymbol{x}_k^l W_V^h + \boldsymbol{b}_V^h) W_O^l + \boldsymbol{b}_O^l \tag{13}$$

となる．(各 Layer 毎に定義される式であることを明示するため，Layer 番号を l で表記する．) ここで，$\boldsymbol{x}_k^l$ が MHA における文脈語 k の入力ベクトル，$\sum_h$ は，現在の Layer のすべての Head に関する和を表す．

ここで，最終層の出力

$$\boldsymbol{o}_{final} = \mathrm{LN}_{final}(\boldsymbol{h}_j^{'''N})$$

と，後続単語の Embedding ベクトル $\boldsymbol{e}_n$ の内積をとり，その softmax を計算し，log をとった式

$$f = \log \frac{\exp(\boldsymbol{e}_n \cdot \boldsymbol{o}_{final})}{\sum(\boldsymbol{e}_j \cdot \boldsymbol{o}_{final})}$$

を計算し，この式を $\boldsymbol{x}_k^l$ の各成分で偏微分する．最急降下法の考え方により，この微分値で得られた各成分によるベクトル

$$\boldsymbol{z}_k^l = < \frac{\partial f}{\partial \boldsymbol{x}_{k1}^l}, \frac{\partial f}{\partial \boldsymbol{x}_{k2}^l}, \cdots, \frac{\partial f}{\partial \boldsymbol{x}_{kn}^l} >$$

は，出力ベクトルを $\boldsymbol{e}_n$ 方向に動かす入力ベクトルとなっていると考えられる．これらの偏微分の値は，誤差逆伝搬法により計算することができる．

5.1 文脈ベクトルからの文脈単語の推定

得られた各 $\boldsymbol{z}_k^l$ が，実際にどの単語に関連が深いのかを測定するため，「文脈語が，FL 法によって出力ベクトルへ変換されている」という仮定をおく．単語 d が文脈語の入力として与えられ，FL 法により出力ベクトルに変換されたとすると，各層における出力ベクトル（および次層への入力ベクトル）は，第 1 層の Head の重みを 1 とした変換[*10]によって得られる．このさいに得られる各層への入力を，$\boldsymbol{h}_l(d)$ として定義する．すなわち，$\boldsymbol{h}_l(d)$ は，「FL 法により単語 d の Embedding ベクトルが出力ベクトルに変換される過程における，層 l への入力ベクトル」である．

$\boldsymbol{z}_k^l$ と各 $\boldsymbol{h}_l(d)$ の内積をとったものは，層 l において，文脈単語 k として単語 d が用いられたとき（文脈単語 $\boldsymbol{x}_k^l$ に単語 d の成分が加算されたとき）の損失関数 f の変化を表していると考えられる．これを第 2 層以降のすべてで和をとった

$$\sum_{l \geq 2} \boldsymbol{z}_k^l \boldsymbol{h}_l(d)$$

を，単語 d の文脈語としての関連度として定義する．この関連度上位の単語を bigram $c.n$ の文脈単語として出力する．[*11]

[*10] FLQK 法においては，第 1 層のうち，自己 QK 類似度がしきい値を超えた Head の重みを 1 とした変換

[*11] なお，FL 法以外の手法については，Head 重みが 0 になるすべての Head について和をとる．これは，我々の提案手法において，「Head 重みが 1 となる Head を用いて bigram の候補が得られ，残りの Head を用いて実際の次単語が得られる」という仮定を置いているためである．

表 1　使用したモデル

model	token	layer	head	hidden	ffn	ratio
waseda-xl	50,000	48	20	1600	6400	8.36
titech	32,771	36	20	1024	4096	45.56
line	51,200	24	24	2304	9216	2.67
abeja	32,000	24	16	1280	5120	38.91
rinna-m	32,000	24	16	1024	4096	18.04
rinna-s	32,000	12	12	1024	3072	2.52
waseda-s	32,000	12	12	768	3072	1.15
rinna-xs	32,000	6	8	512	2304	8.77

6　実験

モデルとして, rinna-m[*12], rinna-s[*13], rinna-xs[*14], waseda-xl[*15], waseda-s[*16], line[*17], titech[*18], abeja[*19] の 8 種類のモデルを用いた.[*20] 1 に各モデルのサイズを示す．ここで，token はトークン数，layer はレイヤー数，head は各 MHA におけるヘッド数，hidden は隠れ状態数，ffn は，FFN 層の第 1 層における射影先の隠れ状態数である．また，ratio については，後述する．

6.1　高頻度語の選定

我々は，コーパスの中から，高頻度でかつ意味のある単語を選定するために，以下の手順で，**高頻度語**の集合 F を選定し，実験に用いた.

1. GPT-2 に登録されているトークンのうち，日本語 WordNet 中の Synset に登録されている単語のみを残す.
2. 手順 1. で選定された単語の上位 1000 個を選ぶ.

[*12] https://huggingface.co/rinna/japanese-gpt2-medium
[*13] https://huggingface.co/rinna/japanese-gpt2-small
[*14] https://huggingface.co/rinna/japanese-gpt2-xsmall
[*15] https://huggingface.co/nlp-waseda/gpt2-xl-japanese
[*16] https://huggingface.co/nlp-waseda/gpt2-small-japanese
[*17] https://huggingface.co/line-corporation/japanese-large-lm-1.7b
[*18] https://huggingface.co/okazaki-lab/japanese-gpt2-medium-unidic
[*19] https://huggingface.co/abeja/gpt2-large-japanese
[*20] なお，各モデルの語彙には Subword も含まれているが，以下では便宜上，Subword についても「単語」と表記する.

6.2 実験 1: 後続単語リストの取得

まず，FL 法によって各クエリ c に対して得られた後続単語リストが，実際に後続しやすい語になっているかについて，実際のコーパスを用いて検証する．

コーパスには，日本語 Wikipedia から取得したテキスト（35,018,927 行）および，日本語 CC-100 から取得したテキスト（45,838,795 行）を用いた．

実験では，コーパスを単語分割し，F 中の各単語について，連接頻度上位語のリストを作成する．各候補語について，頻度上位 N 個の単語を「連接しやすい単語」として正解語集合とし，FL 法で得られた単語リスト上位の語が，これらの正解語集合に含まれているかどうかの精度を測定する．なお，各単語に後続する単語については，F に含まれていない単語も含め，全単語を出力・評価の対象とする．これは，ある単語を特徴づける後続語として，助詞等の機能語も重要であると考えたためである．

結果の評価には，平均精度 [Chakrabarti 02] を用いる．各入力単語 w に対するアルゴリズムの出力リストを $\langle b_1, b_2,$
$..., b_n\rangle$ としたとき，平均精度は，

$$\frac{1}{|S(t)|}\sum_{1\leq k\leq n} r_k \cdot prec(k),$$

と計算される．($prec(k)$ は，上位 k 個を見た時の正解率．また，r_k は，$c_k \in S(t)$（c_k が正解）なら 1，そうでなければ 0 をとる変数.）これは，各正解語の出現において，それより上の順位で出力された語を全出力と見なしたときの正解率（精度）を測定し，その平均をとったものである．

比較対象として，FL 法について，第 1 層ではなく，第 2 層の Head 重みを 1 とした場合 (L= 2) と，第 1 層および第 2 層の Head 重みをすべて 1 とした場合（L≤ 2）についても実験を行った. [*21] [*22]

■結果　2 および 3 に結果を示す．全体的に，Wikipedia よりも CC-100 のほうが高い精度を示したが，手法による精度の違いに関しては，どちらのコーパスでも同じ傾向を示した．

FLQK 法および FL 法については，どのモデルに対しても，安定した精度を示す一方，モデルによっては，MLP 法が高い精度を示す場合が見られた．L=2 の精度は，MLP 法と同様の傾向を示した．パラメータ数の多いモデル（waseda-xl, titech, line, abeja, rinna-medium）のうち line を除く 4 つのモデルで，MLP の精度は FL 法，FLQK 法を大きく下回った．一方，パラメータ数の少ない 3 つのモデル（rinna-s, waseda-s, rinna-xs）については，いずれも MLP 法が FLQK 法や FL 法に遜色ない精度を示した．モデルによって，MLP 層のみで bigram 知識を取り出せるモデルがある一方で，特にモデルサイズの大きい場合は，第 1 層の知識も併用しなければ bigram 知識が取り出せないことがわかった．line については，モデルサイズが大きいにもかかわらず，MLP 法が高い精度を示したほか，第 2 層を併用した場合 (L=2) において最も高い精度を示す等，他のモデルと異なる挙動を示した．

L≤ 2 については，Wikipedia コーパスを用いた際に，一部モデル（waseda-xl の N=10 と 50, titech の N=10, abeja の N=10, waseda-small の N=100 等）において FL 法よりも若干高い精度を示した（表 2）が，

[*21] FLQK 法については，第 1 層を対象としたアルゴリズム（入力が単語 Embedding であることを前提としているため）であるため，FL 法のみを対象としている．

[*22] なお，L= 3（および L≤ 3）についても実験を行ったが，殆どの場合で L= 2（および L≤ 2）の結果を下回ったほか，最高精度となったケースは，model=waseda-xl, L ≤ 3 の場合だけだった（このとき，N=10 において，L ≤ 2 の 0.503 を上回る，0.509 を記録した）ため，省略する．

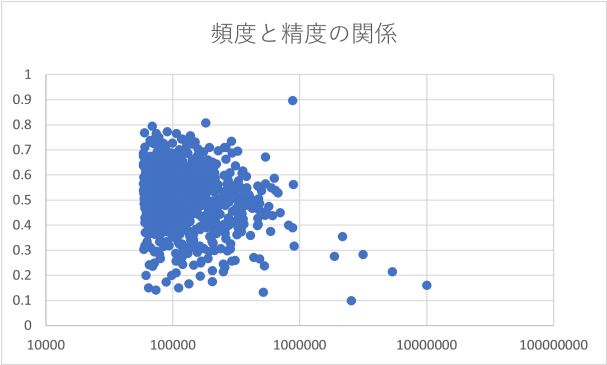

図 3　頻度と精度の関係．相関係数=-0.22

概ね FL 法よりも精度が低下した．これにより，提案手法の仮説通り，第 1 層の Head が次単語予測に重要であり，第 2 層を併用しても，精度への貢献はわずかであるか，逆に悪影響であることが多いことがわかった．

1 の最右列に，第 1 層の MHA 層における残差接続での，各項のベクトルサイズの比率（2 における，$\boldsymbol{h}_j$ と $\boldsymbol{h}_j$ のサイズの比率）を示す．これは，Wikipedia コーパスを用いた実験において用いられた各クエリについて，比率の平均をとったものである．一般的に，大きいモデルについては比率が高い（MHA 層の結果の影響が大きい）結果となった．MHA 層の影響が大きい場合は，MHA 層に bigram 知識が蓄積されている傾向が強いと考えられる．実際に，line においては，比率が小さくなっており，これは，line において MLP 法の精度が高いという結果と整合する．

また，logit lens 法 [nostalgebraist 20] と同様に，第 1 層の出力をそのまま最終層とする設定（以下，direct 法）についても実験した．このさい，FL 法，FLQK 法，MLP 法についてそれぞれ direct 法を実行し，一番精度が高くなったモデルを表中 direct として示す．（どのモデルを用いたかは，モデル名の隣に記述してある．）direct 法については，どのモデルについても，提案手法を下回る精度となった．direct 法による出力は，同義語や関連語の割合が高く，後続単語予測という点では提案手法に劣る結果となった．これは，実際の後続単語空間への射影には，第 2 層以降の MLP 層による変換も重要であることを示していると考えられる．

4 に，FL 法, FLQK 法それぞれについて，提案手法で取得した単語リストの例を示す．（モデルは rinna-m, コーパスは Wikipedia）また，得られた結果についての意味を把握しやすくするため，WordNet に登録された語のみを出力させた結果を，5 に示す．両手法で，得られた単語リストの順位に違いが見られたが，各単語に対し，後続しやすい語が上位に来ていることが観察できる．FL 法のほうが高い精度を示した理由として，FL 法がより一般的な単語を，FLQK 法がよりクエリに特化した単語を重視する傾向があるためではないかと考えられる．

なお，クエリ単語の頻度によって精度に影響があるかどうかを見るために，クエリ単語の頻度と精度の関係を調べた．（モデルは rinna-m, コーパスは Wikipedia, 手法は FL 法.）3 に結果を示す．対数頻度と精度との相関係数は-0.22 となり，明確な相関は見られなかった．

6.3　文脈単語取得手法の評価

文脈単語取得手法では，クエリとして bigram $c.n$ が与えられたとき，「後続語 n の選択に効果のある文脈語」を選択する．この効果を測定するため，bigram のうち，「後続語により意味が変わる bigram」を選ぶ．

表 2　後続単語予測の Average Precision, Wikipedia コーパス.

上位 N	FLQK	FL	L $\leq$ 2	MLP	L=2	direct
model=waseda-xl, direct-FL						
10	0.486	0.495	**0.503**	0.092	0.114	0.261
50	0.484	0.486	**0.487**	0.071	0.083	0.200
100	**0.464**	0.460	0.452	0.055	0.066	0.172
model=titech, direct-FL						
10	0.315	0.549	**0.563**	0.018	0.018	0.149
50	0.301	**0.531**	0.440	0.028	0.029	0.107
100	0.280	**0.481**	0.375	0.036	0.037	0.090
model=line, direct-MLP						
10	0.310	0.325	0.331	0.310	**0.386**	0.103
50	0.404	0.406	0.397	0.404	**0.474**	0.150
100	0.411	0.403	0.387	0.411	**0.464**	0.166
model=abeja, direct-FLQK						
10	0.249	0.306	**0.308**	0.066	0.042	0.188
50	0.309	**0.370**	0.310	0.068	0.042	0.142
100	0.314	**0.374**	0.290	0.059	0.037	0.109
model=rinna-medium, direct-FLQK						
10	0.357	**0.440**	0.426	0.121	0.121	0.074
50	0.460	**0.523**	0.477	0.115	0.114	0.096
100	0.473	**0.516**	0.464	0.095	0.095	0.100
model=rinna-small, direct-MLP						
10	0.342	0.337	0.219	0.343	**0.401**	0.067
50	0.448	0.346	0.230	**0.450**	0.448	0.121
100	0.457	0.322	0.216	**0.459**	0.437	0.146
model=waseda-small, direct-FL						
10	0.174	**0.185**	0.177	0.172	0.175	0.040
50	0.162	**0.173**	0.172	0.160	0.167	0.045
100	0.144	0.156	**0.159**	0.144	0.147	0.049
model=rinna-xsmall, direct-FLQK						
10	**0.285**	0.157	0.077	0.285	0.266	0.047
50	0.263	0.132	0.082	**0.282**	0.260	0.064
100	0.233	0.114	0.062	**0.260**	0.237	0.062

この目的ため，「日本語 WordNet の登録語となっている bigram」を実験対象とし，そのうち頻度上位 1000 個を実験に使用する．例えば，「年表」という単語は，「年」と「表」の単語 bigram であるが，「年表」という bigram そのものも WordNet の登録語である．これにより，各単語の後続語の中で，大きく意味を変化させる後続語を選ぶことができると考えられる．

なお，同一の単語の後続語の影響が大きくなることを防ぐため，同一単語の後続語は最大 3 つまでとした．（例えば，「年」からはじまる bigram を「年月」「年表」「年中」のみとする）

本実験では，文脈語は，F に含まれているものだけを対象とした．これは，助詞等の機能語が，文脈語として後続語に影響を与える可能性が低いと考えたためである．各 bigram と関連性の高い正解文脈語を取得するため，実験 1 と同一のコーパスを用い，F 中の各単語と，各 bigram との同一文での共起頻度を測定し，

表 3　後続単語予測の Average Precision, CC-100 コーパス.

上位 N	FLQK	FL	L≤ 2	MLP	L=2	direct
model=waseda-xl, direct-FL						
10	0.614	**0.618**	0.617	0.063	0.081	0.377
50	**0.581**	0.580	0.570	0.055	0.062	0.281
100	**0.588**	0.581	0.563	0.049	0.056	0.236
model=titech, direct-FL						
10	0.465	**0.676**	0.605	0.015	0.015	0.194
50	0.443	**0,657**	0.501	0.032	0.033	0.157
100	0.417	**0.617**	0.450	0.047	0.048	0.146
model=line, direct-MLP						
10	0.394	**0.418**	0.332	0.394	0.416	0.134
50	0.506	0.512	0.395	0.506	**0.525**	0.210
100	0.533	0.533	0.406	0.533	**0.544**	0.230
model=abeja, direct-FLQK						
10	0.265	**0.465**	0.331	0.035	0.022	0.131
50	0.295	**0.506**	0.349	0.049	0.033	0.124
100	0.300	**0.513**	0.351	0.051	0.034	0.113
model=rinna-medium, direct-FLQK						
10	0.506	**0.531**	0.477	0.062	0.061	0.137
50	0.577	**0.605**	0.530	0.083	0.083	0.134
100	0.595	**0.618**	0.544	0.085	0.085	0.129
model=rinna-small, direct-MLP						
10	0.496	0.375	0.222	**0.498**	0.439	0.130
50	0.570	0.396	0.229	**0.574**	0.504	0.169
100	0.586	0.398	0.231	**0.589**	0.518	0.183
model=waseda-small, direct-FL						
10	0.236	**0.244**	0.219	0.231	0.222	0.036
50	0.178	**0.189**	0.182	0.175	0.181	0.042
100	0.160	**0.171**	0.169	0.159	0.163	0.048
model=rinna-xsmall, direct-FLQK						
10	0.385	0.173	0.054	**0.416**	0.414	0.079
50	0.377	0.154	0.060	**0.418**	0.418	0.078
100	0.379	0.152	0.062	0.415	**0.419**	0.074

Pointwise Mutual Information (PMI) を計算することで，F 中の単語をランク付けした．このランキング上位 N 個の単語を正解とし，提案手法でどの程度上位に正解を出力できたかを，平均精度で測定する．

PMI の計算は，以下の式で行う．ここで，bigram の第一単語を c，第二単語を n とする．

$$pmi_c(n;x) = \log P(n,x|c) - \log P(n|c) - \log P(x|c)$$

すなわち，コーパスにおいて，第一単語 c の出現を前提条件としたときの，後続単語 n の出現確率および文脈語の出現確率をもとに PMI を計算する．これは，本研究での GPT-2 の動作の仮説である，「c のベクトルが，文脈語の影響により n に変化する」という変化を測るためである．

このとき，頻度の測定において，以下の問題が発生した．

表 4　出力の例（後続単語予測）．各単語の頻度ランクは，それぞれ，大:8 位, 高:45 位, 世界:155 位, 最高:807 位.

クエリ	出力上位語
FL 法	
大	根, 好きな, 豆, 福, 好き, 樹, 門, 腸, 黒, 東, 杉
高	円, 輪, 専, 尾, の, は, ?, (, 血圧, 井戸, 品質, を
世界	を, が, 一周, 中, 最速, は, 大会, 最強, 地図, 征服
最高	齢, 経営, に, 級, 級の, 品質, 時速, で, ランク, 潮
FLQK 法	
大	人気, 好きな, 体, 雨, 勢, 田, 丈夫, 豆, 西, 東, 腸
高	円, 品質, 血圧, 畑, 画質, 城, 所, 千, 感度, 尾, 見
世界	が, は, を, には, 貿易, 中, 一周, 最大級の, 最強
最高	に, 経営, で, 時速, 齢, です, 級の, 級, 品質, 潮

表 5　出力の例（後続単語予測, WordNet に存在する語のみ）

クエリ	出力上位語
FL 法	
大	根, 豆, 福, 好き, 樹, 門, 腸, 黒, 東, 杉, 祭, 動脈
高	円, 輪, 尾, 血圧, 井戸, 品質, 収入, 見, 井, 根
世界	一周, 中, 最速, 大会, 地図, 征服, 平和, 樹, 最高
最高	齢, 経営, 級, 品質, ランク, 潮, 額, 傑作, 得点, 値
FLQK 法	
大	人気, 体, 雨, 勢, 田, 丈夫, 豆, 西, 東, 腸, 黒, 好評
高	円, 品質, 血圧, 畑, 城, 所, 千, 感度, 尾, 見, 井
世界	一周, 中, 最速, 大会, 地図, 征服, 平和, 樹, 最高
最高	経営, 齢, 級, 品質, 潮, 傑作, ランク, 額, 値, 記録

1. $P(x|c)$ の計算において，x と c の共起頻度 $freq(x,c)$ が不十分の場合があり，$pmi_c(n;x)$ の値の信頼性が低下する．
2. $P(n|c)$ の値が 1.0 に近い場合（c の後続語の大部分が n である場合），$P(n,x|c)$ と $P(x|c)$ の差が非常に小さくなり，同様に $pmi_c(n;x)$ の信頼性が低下する．

これを避けるため，しきい値 a と b を設定し，$freq(x,c) \geq a$ かつ $P(n|c) < b$ の場合のみを実験対象とした．本実験では，$a = 30, b = 0.8$ とした．

ベースラインとして，各単語に対応する入力 Embedding を用い，各出力単語 n と文脈語の類似度を Embedding の内積で定義する手法を用いる．（表中 base）

6 および 7 に，実験結果を示す．全体の傾向としては，実験 1 と同様の結果となった．すなわち，実験 1 において高い精度を示したモデルが，実験 2 でも高い精度を示した．ただし，FLQK 法と FL 法の比較では，実験 1 では FL 法のほうが高くなるケースが多かったのに対し，実験 2 では，FLQK 法が FL 法を上回る場合が多く見られた．

ベースラインとの比較では，一部（waseda-s モデルで Wikipedia コーパスを用いた場合）でベースラインの精度が一番高くなったものの，殆どのケースで，MLP 法も含めた提案手法が，ベースラインを上回った．「入力単語 c の次単語を n にする文脈単語」の取得というタスクにおいて，提案手法が，単純に次単語 n との

表 6　文脈単語予測の Average Precision, Wikipedia コーパス.

上位 N	FLQK	FL	L≤ 2	MLP	L=2	base
model=waseda-xl						
10	**0.060**	0.057	0.056	0.039	0.039	0.052
50	**0.162**	0.159	0.157	0.132	0.131	0.139
100	**0.284**	0.283	0.279	0.247	0.247	0.251
model=titech						
10	0.041	**0.065**	0.059	0.035	0.035	0.048
50	0.127	**0.162**	0.151	0.116	0.116	0.134
100	0.229	**0.269**	0.256	0.213	0.213	0.229
model=line						
10	0.081	0.077	0.079	0.082	**0.085**	0.061
50	0.201	0.195	0.196	0.201	**0.204**	0.166
100	0.346	0.340	0.340	0.346	**0.348**	0.301
model=abeja						
10	0.079	**0.085**	0.082	0.071	0.070	0.076
50	0.268	**0.275**	0.269	0.256	0.255	0.253
100	0.459	**0.465**	0.456	0.444	0.444	0.427
model=rinna-medium						
10	**0.085**	0.081	0.081	0.063	0.063	0.063
50	**0.272**	0.267	0.265	0.237	0.237	0.228
100	**0.488**	0.483	0.481	0.450	0.450	0.432
model=rinna-small						
10	**0.080**	0.067	0.065	**0.080**	0.078	0.061
50	**0.263**	0.245	0.240	**0.263**	0.259	0.225
100	0.483	0.464	0.460	**0.483**	0.479	0.434
model=waseda-small						
10	0.050	0.053	0.049	0.051	0.051	**0.060**
50	0.081	0.082	0.080	0.081	0.080	**0.086**
100	0.131	0.133	0.131	0.131	0.130	**0.134**
model=rinna-xsmall						
10	0.066	0.064	0.064	**0.068**	0.067	0.063
50	0.241	0.235	0.235	**0.243**	0.241	0.225
100	0.449	0.442	0.438	**0.451**	0.449	0.420

類似度を取得するよりも，概ね高い精度を示すことがわかった.

8 に，FLQK 法で取得した上位文脈単語リストの例を示す.「年表」に対する「歴史」等，bigram の構成語だけからでは推測しづらい文脈語が推定できていることが観察できた.

7　おわりに

本稿では，大規模言語モデルの分析の方向性として，言語モデルのパラメータをマイニングすることにより，モデルに蓄えられた言語知識を取り出すという方針を提唱し，その具体的な例として，GPT-2 から bigram 知識を取り出す手法を提案した．提案手法では，MLP 層および Layer-1 の MHA 層を利用することにより，周

表 7　文脈単語予測の Average Precision, CC-100 コーパス.

上位 N	FLQK	FL	L≤ 2	MLP	L=2	base
model=waseda-xl						
10	**0.065**	0.059	0.055	0.027	0.027	0.046
50	**0.143**	0.133	0.129	0.090	0.090	0.101
100	**0.234**	0.224	0.220	0.170	0.170	0.178
model=titech						
10	0.040	**0.079**	0.066	0.023	0.023	0.040
50	0.096	**0.147**	0.129	0.075	0.075	0.088
100	0.166	**0.225**	0.203	0.139	0.140	0.151
model=line						
10	0.095	0.092	0.094	0.095	**0.100**	0.078
50	0.343	0.337	0.336	0.343	**0.346**	0.301
100	**0.559**	0.555	0.551	0.559	0.559	0.509
model=abeja						
10	0.042	**0.072**	0.059	0.026	0.026	0.040
50	0.106	**0.148**	0.130	0.086	0.086	0.099
100	0.183	**0.232**	0.212	0.161	0.160	0.177
model=rinna-medium						
10	**0.105**	0.094	0.096	0.032	0.032	0.055
50	**0.185**	0.167	0.168	0.093	0.093	0.106
100	**0.273**	0.252	0.252	0.166	0.166	0.176
model=rinna-small						
10	0.106	0.057	0.049	**0.107**	0.103	0.056
50	0.181	0.121	0.105	**0.182**	0.175	0.107
100	0.267	0.200	0.179	**0.269**	0.259	0.175
model=waseda-small						
10	0.097	**0.097**	**0.097**	0.097	0.097	0.089
50	0.231	0.229	**0.231**	0.231	0.231	0.209
100	0.280	0.278	0.280	**0.280**	0.280	0.254
model=rinna-xsmall						
10	0.039	0.034	0.034	**0.056**	0.056	0.044
50	0.103	0.092	0.093	**0.125**	0.121	0.098
100	0.182	0.165	0.166	**0.205**	0.200	0.171

辺単語の情報を使わずに，入力単語の後続語集合を得ることができた．このことは，Layer-1 の MHA 層と，各 Layer の MLP 層が，bigram 知識の抽出に重要であることを示唆している．また，各 bigram に関連する文脈語を，誤差逆伝搬法を用いることによって取得する手法も提案した．本研究提案により，GPT-2 が入力単語から，文脈語の情報を取り入れ，後続語を出力する過程について，一定の示唆を与えられたと考えられる．

今後の課題としては，より多様なモデルでの本手法の適用，bigram ではなく単語をクエリとした関連語抽出への応用，等が挙げられる．また，複数の単語（フレーズ）を入力とした後続単語の取得など，trigram 以上の知識の抽出も今後の課題である．

表 8　出力の例（文脈単語予測）

クエリ	出力上位語
年/表	歴史, 表, 史, 図, 裏, 科学, 構成, 世紀, 構造, 記録
1/番	一番, 番, 前に, 最も, 前の, としては, のような
ス/ラム	ラム, 大統領, 戦争, 陸軍, 州, 帝国, 世紀, 国家, ビル
人/造	宇宙, 製造, 地球, 建設, 工業, ダム, エンジン, 細胞
人/柄	好, 会長, 性格, 社長, 優, 和, 政治, 勝, 忠, 良, 仲
市/電	電気, 鉄道, 駅, パリ, 移動, 交通, 空港, ホテル, 列車
日本/酒	酒, 料理, 魚, 塩, 夜, 食, 刀, 水, 味, 男, 温泉, 蔵
長/調	調, 曲, 作曲, cd, 音楽, 演奏, 楽曲, 編曲, 薬, 合, 状

8　謝辞

本研究は JSPS 科研費 JP20K12027, JP21K12141, JP24K15193 の助成を受けたものです.

参考文献

[Abnar 20] Samira Abnar, Willem H. Zuidema: Quantifying Attention Flow in Transformers. ACL 2020: 4190-4197

[Baeumel 23] Tanja Baeumel, Soniya Vijayakumar, Josef van Genabith, Guenter Neumann, Simon Ostermann: Investigating the Encoding of Words in BERT's Neurons using Feature Textualization. Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP, pages 261?270, 2023.

[Belrose 23] Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, Jacob Steinhardt: Eliciting Latent Predictions from Transformers with the Tuned Lens. CoRR abs/2303.08112, 2023.

[Bills 23] Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, William Saunders? Language models can explain neurons in language models, URL: https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html, 2023.

[Bricken 23] Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nicholas L Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, Chris Olah, Towards Monosemanticity: Decomposing Language Models With Dictionary Learning URL: https://transformer-circuits.pub/2023/monosemantic-features/index.html

[Brown 20] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M.

Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, Dario Amodei: Language Models are Few-Shot Learners. NeurIPS 2020.

[Chakrabarti 02] Soumen Chakrabarti, Mining the Web : Discovering Knowledge from Hypertext Data, Morgan-Kaufmann Publishers, 2002.

[Clark 19] Kevin Clark, Urvashi Khandelwal, Omer Levy, Christopher D. Manning: What Does BERT Look At? An Analysis of BERT's Attention. BlackboxNLP@ACL 2019: 276-286.

[Dar 23] Guy Dar, Mor Geva, Ankit Gupta, Jonathan Berant: Analyzing Transformers in Embedding Space, ACL (1) 2023: 16124-16170

[Devlin 19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL-HLT (1) 2019: 4171-4186

[Ferrando 22] Measuring the Mixing of Contextual Information in the Transformer Javier Ferrando, Gerard I. Gallego, Marta R. Costa-jussa, EMNLP 2022, 8698 - 8714.

[Geva 21] Mor Geva, Roei Schuster, Jonathan Berant, Omer Levy: Transformer Feed-Forward Layers Are Key-Value Memories. EMNLP (1) 2021: 5484-5495

[Geva 22] Mor Geva, Avi Caciularu, Kevin Ro Wang, Yoav Goldberg: Transformer Feed-Forward Layers Build Predictions by Promoting Concepts in the Vocabulary Space. CoRR abs/2203.14680 (2022)

[Gupta 21] Ankit Gupta, Jonathan Berant: Value-aware Approximate Attention. EMNLP (1) 2021: 9567-9574

[Gurnee 23] Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, Dimitris Bertsimas: Finding Neurons in a Haystack: Case Studies with Sparse Probing. CoRR abs/2305.01610 (2023)

[Held 23] William Held, Diyi Yang: Shapley Head Pruning: Identifying and Removing Interference in Multilingual Transformers. EACL 2023: 2408-2419

[Kobayashi 20] Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, Kentaro Inui: Attention is Not Only a Weight: Analyzing Transformers with Vector Norms. EMNLP (1) 2020: 7057-7075

[Kobayashi 21] Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, Kentaro Inui: Incorporating Residual and Normalization Layers into Analysis of Masked Language Models. EMNLP (1) 2021: 4547-4568

[Koto 21] Fajri Koto, Jey Han Lau, Timothy Baldwin: Discourse Probing of Pretrained Language Models. NAACL-HLT 2021: 3849-3864

[Langedijk 23] Anna Langedijk, Hosein Mohebbi, Gabriele Sarti, Willem H. Zuidema, Jaap Jumelet: DecoderLens: Layerwise Interpretation of Encoder-Decoder Transformers. CoRR abs/2310.03686 (2023)

[Lin 19] Yongjie Lin, Yi Chern Tan, Robert Frank: Open Sesame: Getting inside BERT's Linguistic Knowledge. BlackboxNLP@ACL 2019: 241-253

[Marecek 19] David Marecek, Rudolf Rosa: From Balustrades to Pierre Vinken: Looking for Syntax in Transformer Self-Attentions. BlackboxNLP@ACL 2019: 263-275

[Michel 19] Paul Michel, Omer Levy, Graham Neubig: Are Sixteen Heads Really Better than One? NeurIPS 2019: 14014-14024

[nostalgebraist 20] nostalgebraist. URL: https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-

gpt-the-logit-lens, 2020.

[Oba 21] Daisuke Oba, Naoki Yoshinaga, Masashi Toyoda: Exploratory Model Analysis Using Data-Driven Neuron Representations. BlackboxNLP@EMNLP 2021: 518-528

[Reif 19] Emily Reif, Ann Yuan, Martin Wattenberg, Fernanda B. Viegas, Andy Coenen, Adam Pearce, Been Kim: Visualizing and Measuring the Geometry of BERT. NeurIPS 2019: 8592-8600

[Vaswani 17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin: Attention is All you Need. NIPS 2017: 5998-6008

[Vig 19] Jesse Vig, Yonatan Belinkov: Analyzing the Structure of Attention in a Transformer Language Model. BlackboxNLP@ACL 2019: 63-76

[Voita 19] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, Ivan Titov: Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned. ACL (1) 2019: 5797-5808

[Voita 19] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, Ivan Titov: Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned. ACL (1) 2019: 5797-5808

[Sakarvadia 23] Mansi Sakarvadia, Arham Khan, Aswathy Ajith, Daniel Grzenda, Nathaniel Hudson, Andre Bauer, Kyle Chard, Ian T. Foster: Attention Lens: A Tool for Mechanistically Interpreting the Attention Head Information Retrieval Mechanism. arXiv:2310.16270v1, 2023.

[Radford 19] Alec Radford, Jeff Wu, Rewon Child, D. Luan, Dario Amodei, Ilya Sutskever, Language Models are Unsupervised Multitask Learners, URL: https://www.semanticscholar.org/paper/Language-Models-are-Unsupervised-Multitask-Learners-Radford-Wu/9405cc0d6169988371b2755e573cc28650d14dfe, 2019.

[Radford 18] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, Improving Language Understanding by Generative Pre-Training, URL: https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf, 2018.

[吉田 23] 吉田 稔, 松本 和幸, 北 研二, Value 行列を手掛かりとした Transformer の分析, 人工知能学会論文誌, Vol.38, No.2, 2023 年.