生成AIによる言語モデルの紹介

–デジタルミュージアム・人文学研究会における視点–

大野 邦夫 †

†株式会社 モナビIT コンサルティング

†E−mail: k−ohno@star.ocn.ne.jp

あらまし　生成AIに関しては,多くの雑誌や書籍が出版され社会的な議論になっているが ,その言語モデルに関する数学的な技術情報はあまり扱われていない .だが技術的な検討を進めるために ,基礎となる数学モデルの把握が必要と思われる .本稿はその試みとして ,Word2vec, RNN, LSTM, ELMo, Transformerに関する統計数学的な手法を紹介すると共に, 固有表現抽出技術に関して既存のコンテンツ管理技術との関係を考察する .

キーワード　生成AI, 言語モデル, Word2vec, RNN, LSTM, ELMo, Transformer, 固有表現抽出

An Introduction to Language Models based on Generative AI

− In View of SIG on Digital Museum and Humanities−

Kunio Ohno †

†Monavis IT Consulting Co.

†E−mail: k−ohno@star.ocn.ne.jp

Abstract　Regarding generative AI, many magazines and books have been published and there is a lot of social discussion, but there is not much mathematical technical information about the language models. In order to proceed with technical considerations, it seems necessary to understand the underlying mathematical models. Then, this paper introduces statistical mathematical methods for Word2vec, RNN, LSTM, ELMo, and Transformer, and considers the relationship between named entity extraction technology and existing content management technology

Keywords　Generative AI, Language model, Word2vec, RNN, LSTM, ELMo, Transformer, Named entity extraction technology

1. はじめに

3年前の年次大会で,計算機によるメタデータ抽出に関して考察したが[1],その際にWord2vecやBERTに関する技術について簡単に紹介した. その内容は, BERTに関する論文[2]の理解に基づく紹介であったが, その後 BERTを活用するためのチュートリアルを兼ねた専門書 [3]が出版され, 統計数学に基づく言語モデルのアルゴリズムが詳細に解説されており, 興味深く思った.

ところで昨年11月にOpenAIのChatGPTが紹介され, この技術が急速に話題になり普及しつつある . そのインパクトは測り知れない感があり , 多くの技術者・研究者にとって必須のものとなりつつある.　　それにも関わらず, ChatGPT関連で出版されている雑誌や書籍の多くは ,その活用やビジネス展開に関するものが大半で , 背景技術内容に関するものは限られているのが現状と感じている . そこで,先に述べた専門書の内容を ,数学モデルにフォーカスして解説することを考えた.

以上のような趣旨でまとめたのが, 本報告である. 2章で対象とする生成AIによる言語モデルの概要について述べ, 3章で文脈非依存のモデルである Word2vecを紹介する.4章で文脈依存のELMoについて述べるが, 併せてRNN, LSTMについても解説する.5章で最新技術であるTransformerについて紹介し,6章で既存コンテンツ技術やデータベース技術に関連すると思われる固有表現抽出について述べる .

2. 生成AIによる言語モデル

2.1 自然言語処理

自然言語処理は, 人間が日常的に使っている言語を計算機が処理する技術である .コンピュータが発明されて以来, プログラミング言語との対比から継続的に研究されてきた課題である.最近の生成AIが話題になって以降は,主に下記のような技術が関係者に注目されているようである .

- 形態素解析：単語を抽出し品詞や活用形を与える
- 言語モデル：文章の自然さを確率によって評価する数理モデル
- 固有表現抽出：人名, 組織名, 日時, 場所, 数値などの属性を与える
- 文章の類似度比較：文章間の意味・内容の類似度を評価するタスク
- 文章分類：文章をカテゴリに分類するタスク
- 文章生成：特定文章に続く文章や , 与えられた条件を満たす文章を生成するタスク
- 文章校正：表記誤りを修正するタスク

本稿では,以上の言語モデルを中心に扱うが,従来のコンピュータ言語との関連が深い固有表現抽出についても簡単に述べる.

2.2 トークン化

文の基本構成要素をトークンと呼ぶ.トークン化は,文を適当な単位に分割する処理でトークナイザと言うツールが用いられ,その際にトークンにはIDが割り当てられる.

システムが受け容れる語彙容量には制約があるので,受け入れられないトークンは未知語と呼ばれ無視されるか未知語IDとして登録される.未知語は極力少なくすることが望まれる.分割方法には,単語単位,文字単位,サブワード単位が挙げられる.

2.3 言語モデル

言語モデルは,文章の出現しやすさを確率で定義するモデルである.別の言い方をすると,文章の自然さを確率で表現する.ある文章,Sをトークン化して得られた結果を$(w_1, w_2, \ldots, w_n)$とすると,文章Sの出現確率は以下のようになる.

$$p(S) = p(w_1, w_2, \ldots, w_n) \quad \cdots\cdots (1)$$

あるトークンの出現確率は,それ以前のトークンに依存するので

$$\begin{aligned} & p(w_1, w_2, \ldots, w_n) \\ & = p(w_1) \times p(w_2|w_1) \times p(w_3|w_1, w_2)\cdots \\ & = \prod_{i=1}^{n} p(w_i|c_i) \end{aligned} \quad \cdots\cdots (2)$$

ここでc_iはw_iを予測する際の前提条件で,この場合は,w_iより前のトークン列$c_i = (w_1, w_2, \ldots, w_{i-1})$である.この$c_i$は,文脈(context)と呼ばれる.すなわち,文章の出現確率をモデル化するには,ある文脈下でのトークンの出現確率である$p(w_i|c_i)$をモデル化すれば良いことが分かる.このことから「言語モデルは,ある文脈下で出現するトークンを予測するモデルである」とも言える.なお文脈という語彙は,一般語でも頻繁に用いられるので,この用語の専門用語としての使用並びに把握には注意が必要である.

自然言語処理のタスクの多くは,与えられたデータに対して,複数のカテゴリに対応付ける分類問題として扱うことが可能である.ここではカテゴリーの数をN,データを表現するベクトルをxとし,分類カテゴリを,1からNの整数で表現し,そのラベルをlとする.分類問題のゴールは,データxが与えられた場合に,そのラベルlを的確に予測する分類モデルを作成することである.

データxを入力した際のニューラルネットワークの出力を,$y = F(x, \theta)$とする.ここでθはニューラルネットワークが持つ各ノードのパラメータである.yはカテゴリーの数の次元のベクトルである.出力ベクトルyの各要素の値は,各カテゴリーへの予測確度を表わし,その値は分類スコアと呼ばれる.通常は分類スコアの最も高いカテゴリーがニューラルネットワークの予測となる.

精度の高い予測を行うには,データからモデルの学習を行う必要がある.モデルの出力yと実際のラベルlとの相違を,損失関数$L(y, l)$で表す.学習に用いるデータセットを用いて損失を計算し,その平均値

$$L = \frac{1}{m} \sum_{i=1}^{m} L(y_i, l_i) \quad \cdots\cdots (3)$$

が小さくなるようにパラメータを決めることが要件になる.ここで,y_iとl_iは,i番目のデータに対する出力と対応ラベルで,mはデータ数である.損失関数を定義するには,分類スコアを予測確率に変換する必要がある.そのために,SoftMax関数を用いて,y_iを予測確率p_iに変換する.

$$p_i = \frac{\exp(y_i)}{\sum_{i=1}^{N} \exp(y_i)} \quad (i = 1, 2, \cdots \quad N) \quad \cdots\cdots (4)$$

この確率データがi番目のカテゴリーに属す時,

$$L(y, l) = - \log p_i \quad \cdots\cdots (5)$$

を分類問題の損失と定義する.これをクロス・エントロピー損失と呼ぶ.

3. 文脈非依存の処理

3.1 Word2vec

分散表現とは,単語の意味概念を数値ベクトルで統計的関係としてに表現することであるが,Word2vecの手法がその典型例である.単純な分散表現は,単語(word)としてのトークンに対して文脈非依存の表現を与えるが,Word2vecは,文字通り単語をベクトルとして処理して意味概念を検討するためのツールであり,分散表現の基本的ツールとして位置づけられる.

単語に文脈非依存の分散表現を与えることを単語埋め込み(Word Embedding)と呼び,意味的類似性を提示するだけでなく,意味的な加算や減算を可能にする.Word2vecには,CBOW(Continuous Bag of Words)とスキップグラム(Skip gram)という二つのモデルが存在する.

3.2 CBOW

CBOWモデルは,文章$S = (w_1, w_2, \cdots \quad w_n)$が与えられた時に,その$i$番目に位置する単語$w_i$をその周囲の単語群$C_i = (w_{i-c}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+c})$から予測する言語モデルである.文脈$C_i$は,$w_i$の前後一定範囲であり,範囲を示すパラメータ$c_i$は,ウインドウサイズと呼ばれる.CBOWは,文脈中の単語をベクトルに変換する埋め込み層V_cと,予測する単語をベクトルに変換する埋め込み層V_tの2つの層から構成されるモデルである.なおV_cによる単語wのベクトルを$v_c(w)$,V_tによるwのベクトルを$v_t(w)$で表す.CBOWでは,文脈中の単語w_iに対するベクトル$v_c(w_i)$を平均することで,文脈を表現し,確率分布$p(w_i|C_i)$をモデル化する.この値を$v(C_i)$とすると,

$$v(C_i) = \frac{1}{2c} \sum_{w_j \in C_i} v_c(w_j) \quad \cdots\cdots (6)$$

CBOWは,この値と予測対象の単語w_iに対応するベクトルの内積により出現し易さを計算する.すなわち,文脈C_iに

おいて語彙中のj番目の単語　s_j　の出現し易さは,$v(C_i)\cdot v_t(s_j)$でスコア付けされる．これをSoftMax関数で確率に変換する．

$$p(w_i = s_j|C_i) = \frac{\exp(v(C_i)\cdot v_t(s_j))}{\sum_{k=1}^{N}\exp(v(C_i)\cdot v_t(s_k))} \quad \cdots (7)$$

式(7)の計算量は,語彙数Nに比例する.そのために巨大な語彙を扱う学習では膨大な計算量になりボトルネックになるのであるが,計算機の進歩がそれを可能にした.学習済のCBOWから単語の分散表現を得るには,埋め込み層V_cを用いる.単語wに対する分散表現は,$v_c(w)$で与えられる．

3.3 Skip−Gram

Skip−Gramは,文章中の対象の単語から周辺の単語を予測する手法で周囲から該当単語を予測するCBOWとは逆である．　文章中の単語w_iが与えられた際に,文脈C_iに出現する単語wの確率分布$p(w|w_i)$をモデル化する手法である．w_iが与えられた際に,文脈中のj番目の単語s_jに対する確率は,以下のように与えられる．

$$p(s_j|w_i) = \frac{\exp(v_c(w_i)\cdot v_t(s_j))}{\sum_{k=1}^{N}\exp(v_c(w_i)\cdot v_t(s_k))} \quad \cdots (8)$$

Word2vecは,文脈非依存なので単語の順序関係は無視して,一意に分散表現を提供するが,そのために多義語の区別が不可能である．その解決のために考案されたのが次に述べるELMo（EmbeddingsfromLanguageModels）である．

4. 文脈依存の処理

4.1 ELMo

Word2vecと異なり,ELMoは文脈に応じて異なる分散表現を得る．この方式を文脈化単語埋め込み（Contextualized Word Embedding）と称する．そのために,Word2vecでは不可能であった同一表現の異なる意味の単語を識別することができる．深層学習方式として,再帰型ニューラルネットワーク（RNN:RecurrentNeuralNetwork）の改良型のLSTM（Long−Short Term Memory）を用いる.そのために,先ずはRNNから説明する．

4.2 RNN

語彙や単語のトークンデータ$(w_1, w_2, \ldots, w_n)$を考え,i番目の出力　h_iは,その時点の入力w_iと,その前のRNN出力h_{i-1}を用いて下記のように表される．

$$h_i = \phi(Aw_i + Bh_{i-1} + b)\ \ (i = 1, 2, \ldots, n) \quad (9)$$

ここでϕはニューラルネットワークの変換関数（活性化関数）,AとBは行列,bはベクトルである.初期値h_0は適当なベクトルとする.式の簡略化のために,式(9)の右辺を単に$\phi(w_i, h_{i-1})$と表すことにする.そうすると式(9)は以下のように再帰的に展開される．

$$\begin{aligned} h_i &= \phi(w_i, h_{i-1}) \\ &= \phi(w_i, \phi(w_{i-1}, h_{i-2})) \\ &= \phi(w_i, \phi(w_{i-1}, \phi(w_{i-2}, h_{i-3}))) \\ &\cdots \end{aligned} \quad \cdots (10)$$

この展開をh_0が出現するまで繰り返すと,i番目の出力h_iは,h_0および$w_1, w_2, \ldots, w_n$の関数として表現できる．

$$h_i = h_{i-1}(w_1, w_2, \ldots, w_{i-1}) \quad \cdots (11)$$

RNNの出力h_{i-1}を線形変換を施してトークンと同じ次元のベクトルh'_{i-1}に変換することが可能である.文脈を考慮したw_iの出現確率　$p(w_i|w_1, w_2, \ldots, w_{i-1})$は,　ベクトル$h'_{i-1}$をSoftMax変換して得られる．

$$p(w_i|w_1, w_2, \ldots, w_{i-1}) = \frac{\exp(h'_{i-1,j})}{\sum_{k=1}^{N}\exp(h'_{i-1,j})} \quad \cdots (12)$$

RNNによる処理モデルを図1に示す．

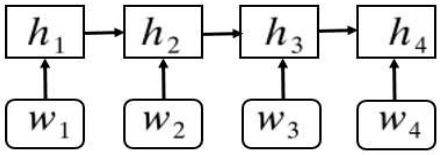

図1　RNNによる処理モデル

以上は,ある位置よりも前にある文章から該当するトークンを予測したが,その位置よりも後ろの文章から予測することも可能である.この場合は,図2のようにRNNを後ろから逆に逐次処理することになる．

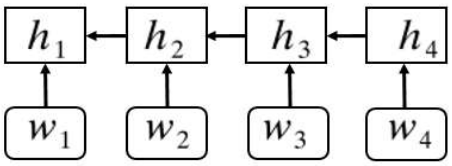

図2　文の後方からのRNNによる処理モデル

RNNは,　図3のように複数層を重ねて処理を行うことが可能で,このような方式を多層RNNと呼ぶ.1層目のRNNは

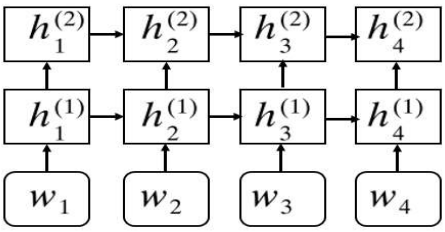

図3 多層RNNによる処理モデル

,入力w_iと同じ層の一つ前の出力の$h_{i-1}^{(1)}$を受けて$h_i^{(1)}$を出力する．2層目の$h_i^{(2)}$は,入力として1層目の出力$h_i^{(1)}$と,2層目

の一つ前の出力の$h_{i-1}^{(2)}$を受けて出力する．k層目のRNNでは，$(k-1)$層目の出力$h_i^{(k-1)}$とk層目の一つ前の出力の$h_{i-1}^{(k)}$を受けて$h_i^{(k)}$を出力する.式で表すと下記のようになる.

$$h_i^{(1)} = \phi(A^{(1)}w_i + B^{(1)}h_{i-1}^{(1)} + b^{(1)})$$

$$h_i^{(2)} = \phi(A^{(2)}h_i^{(1)} + B^{(2)}h_{i-1}^{(2)} + b^{(2)})$$

$$\cdots$$

$$h_i^{(k)} = \phi(A^{(k)}h_i^{(k-1)} + B^{(k)}h_{i-1}^{(k)} + b^{(k-1)}) \quad (13)$$

RNNの各々のトークンにおける出力は,そのトークンの文脈をを考慮した分散表現と見なすことが可能である．この点に関しては,Word2vecの一意的な分散表現とは異なるのである.

4.3 LSTM

RNNは，入力データの増大に伴い，初期のデータの比重が減少することになる.そのために的確な重みを付与させ，データを効率的に処理する手法が検討され，それを実現したのがLSTMである.

LSTMは，RNNの出力に3種類のゲートを付与したシステムとしてモデル化可能である.各ゲートは,対応する入出力値に対して重み付けを行うが，そのためにメモリセルというh_iに相当するデータを一時的に蓄積記憶するc_iという機能を有する．　RNN出力は，　式(13)のように，$h_i = \phi(Aw_i + Bh_{i-1} + b)$となるが，下記の式により，重みベクトル g_I, g_F, g_Oを対応付ける.

$$g_I = \sigma(A_I w_i + B_I h_{i-1} + b_I) \quad \cdots\cdot(14)$$

$$g_F = \sigma(A_F w_i + B_F h_{i-1} + b_F) \quad \cdots\cdot(15)$$

$$g_O = \sigma(A_O w_i + B_O h_{i-1} + b_O) \quad \cdots\cdot(16)$$

σは活性化関数で,要素ごとにシグモイド関数を作用させる.その結果，c_iとh_iは,下記のように計算される.

$$c_i = g_I \odot h_i + g_F \odot c_{i-1} \quad \cdots\cdot(17)$$

$$h_i = g_O \odot \tanh(c_i) \quad \cdots\cdot(18)$$

ここで，$\odot$ は要素ごとの積演算を意味する.なお，重みベクトルg_Iは,RNNの出力値h_iに対して入力データw_iを動的に調整し，Input Gateと呼ばれる．g_Fは，メモリセルの既存データc_{i-1}を乗じて過去に入力されたデータを再評価する機能を有し，Forget Gateと呼ばれる．g_Oは,現状の出力値h_iを計算する際に，次項の計算に適用すべき割合を評価する機能を有し，Output Gateと呼ばれる.

4.4 ELMoの実装

LSTMもRNN同様に順方向と逆方向による予測が可能であるELMoは双方向のLSTMを活用するシステムとして位置づけられる．ELMoの構成を図4に示す．先ず大量の文書を用いて汎用的な言語のパターンを学習させるが，これを事前学習と呼ぶ.

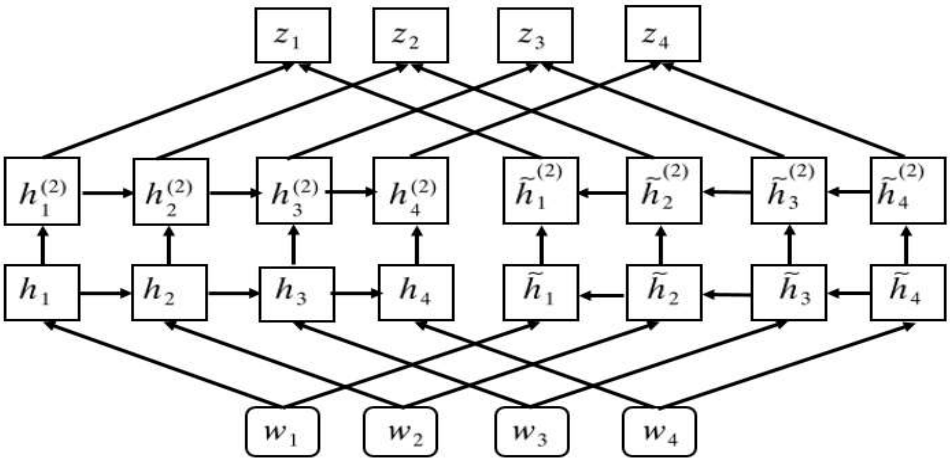

図4 ELMoの処理モデル

適当な文$(w_1, w_2, \ldots, w_n)$が与えられたとき,i番目に位置するトークンw_iの出現確率を予測する．　順方向の多層LSTMからは，w_iより前の文章$(w_1, w_2, \ldots, w_{i-1})$を文脈とする確率$p(w_i|w_1, w_2, \ldots, w_{i-1})$が得られる.他方,逆方向の多層LSTMからは，$(w_{i+1}, w_{i+2}, \ldots, w_n)$を文脈とする確率$p(w_i|w_{i+1}, w_{i+2}, \ldots, w_n)$が得られる．この双方の予測を組み合わせると，損失関数は，式(5)から下記のようになる.

$$L = -\sum_{i=1}^{n} (\log p(w_i|w_1, w_2, \ldots, w_{i-1}) + \log p(w_i|w_{i+1}, w_{i+2}, \ldots, w_n)) \quad \cdots\cdot(19)$$

対象となるトークンに対して,前方の文章,後方の文章双方の予測精度が向上するように学習が行われる．従って,ELMoは，学習を通じて予測対象のトークン以外の全てを文脈とする予測を行っていると言える.

図4は,簡略化した概念モデルであるが,w_1～w_4のトークンは,順方向と逆方向の双方のLSTM処理が行われ,処理結果は，　Z_1～Z_4ほ分散表現データとして格納される.Word2vecでは,一意的なデータとして扱われた分散表現が，

文中の個別のトークン毎に定義されるので，文脈依存となるELMoとWord2vecの規模の違いが認識できる．

5. Transformer

5.1 Transformerの特徴

RNNもLSTMも，それぞれの層の中で文章の前方又は後方から逐次的に処理が行われ，処理を経るにつれ，初期に行った処理結果が失われてしまう．この問題を解決するために考案された技術がTransformerである．

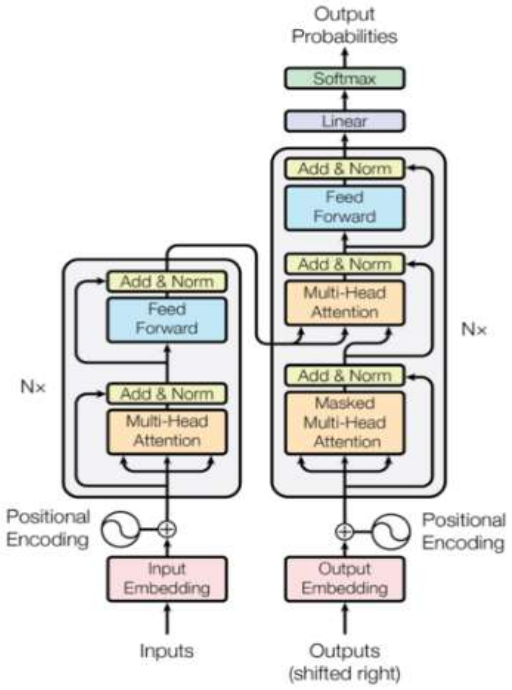

図5 文献[4]で紹介されたTransformer の構成

Transformerを最初に取り上げたのは，“Attention is all you need” [4]という論文である．その論文で紹介されたTransformerの構成を図5に示す．

オープンAIのGPT並びにグーグルのBERTはこの技術を用いて，RNNやLSTMの制約を乗り超えた．

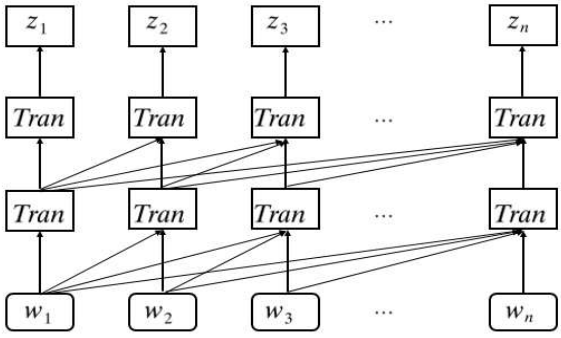

図6 Open AIによるGPTの処理モデル

Transformerが文章をトークンに分割して入力し，ベクトルとして出力する手法に関しては，RNNやLSTMと同様であるが，Transformerでは各々の層が個別のトークン毎に対応するベクトルを出力する．その次の層はそれを受けてさらに新たなベクトルを送り出し，それを繰り返すことに特徴がある．オープンAIのGPTもグーグルのBERTも同様な手法を用いていると思われるが，GPTが単方向だけの処理なのに対して（図6），BERTは双方向の処理を行っている点が基本的な相違である（図7）．

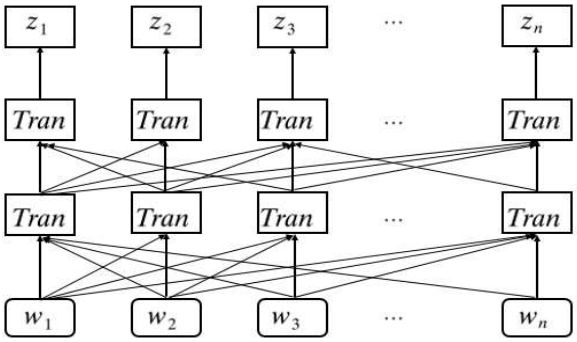

図7 GoogleによるBERTの処理モデル

5.2 Attention

Transformerは，トークン情報を扱う際に，他のトークンの情報を直接参照して処理を行う．この際に，それぞれのトークンの情報にどの程度の注意（Attention）を払うかは，出現するトークンに応じて異なり，適応的に定められる．この方式はAttention（注意機構）と呼ばれ，Transformer処理の特徴とされる．人間が意味的な語彙の重要性を前後の文章や文意から判断することを，言語モデルに行わせるアルゴリズムと考えれば良いであろう．Attentionを用いることにより，離れた位置におけるトークンの情報も重要度に応じて適切に採り入れることができる．このメカニズムは，Transformer Encoderと呼ばれる．

図6，図7ともに，w_1～w_nのトークン列が，Transformer Encoderを通じて相互に参照されてz_1～z_nの分散表現として処理される状況を示している．Transformer Encoderの個々の層は，Multi−Head AttentionとFeedforward Networkから構成される．Multi−Head Attentionは，当初の単一対象の注意機構のAttention機能（Scaled Dot−Product Attention）を，複数の語彙に拡張したことによりその名称が与えられている．

Transformer の内部では，各トークンはベクトルとして表現される．そのために，個別にシーケンシャルに処理されるのではなく，行列として一括して処理される．

5.3 Scaled Dot−Product Attention

Transformer Encoderに含まれる Multi−Head Attention を理解するには，その構成技術であるScaled Dot−Product Attentionを理解する必要がある．Scaled Dot−Product Attentionは，入力検索情報のエンコーダと参照対象情報のメモリーデータで構成される．メモリーデータは，キー（key）と値（value）に区分される．他方エンコーダ情報は照会（query）情報となる．query情報が，key情報を

参照し,さらにその値の情報との関係を用いてAttention機能が実現される.

ここでは先ず, n個のトークンで構成される文章を処理することを想定する.前の層のi番目のトークンに対応する出力は, ベクトルx_i で与えられる$(i = 1, 2, \ldots, n)$. このベクトルを線形変換のための行列 W^Q, W^K, W^Vでクエリ（q_i）,キー（k_i）,値（v_i）の3種のd次元ベクトルに変換する.

$$q_i = x_i W^Q \quad \cdots\cdot (20)$$

$$k_i = x_i W^K \quad \cdots\cdot (21)$$

$$v_i = x_i W^V \quad \cdots\cdot (22)$$

Scaled Dot−Product Attentionにおいて, 個々のトークンは以上の3種類のベクトルで特徴付けられる. Scaled Dot−ProductAttentionは,以上のベクトルを入力として受け,夫々のトークンに対してベクトル a_iを出力する.a_iは,値v_iの重み付き平均で,

$$a_i = \sum_{i=1}^{n} \alpha_{i,j} v_i \quad \cdots\cdot (23)$$

で与えられる. なお重み$\alpha_{i,j}$は, $\alpha_{i,j} \geqq 0$かつ $\sum_{j=1}^{n} \alpha_{i,j} = 1$を満たすことが要件である. 重み$\alpha_{i,j}$は, i番目のトークンを処理する際にj番目のトークンの情報を評価するための重みパラメータである.重みはキーとクエリにより決定される. 評価手法としては, i番目のトークンのクエリq_iとj番目のトークンのキーk_jとの関連度を何らかの方法で評価し, それに応じて重みを決定する. 従って, i番目のトークンを処理する際には, クエリと関連度が大きなキーを持つトークンの影響が大きくなる. TransformerではScaled Dot−Product と呼ばれる方法でクエリとキーを評価する. この手法では, q_iとk_jの内積を$\sqrt{d}$で割って得られる$\tilde{\alpha}_{i,j}$が評価スコアとして用いられる.

$$\tilde{\alpha}_{i,j} = \frac{q_i \cdot k_j}{\sqrt{d}} \quad \cdots\cdot (24)$$

次元dが大きくなると内積は極めて小さくなって計算困難になるので, $\sqrt{d}$で割って計算可能としている. スコア$\tilde{\alpha}_{i,1}, \tilde{\alpha}_{i,2}, \ldots, \tilde{\alpha}_{i,n}$にSofMax関数を適用することにより最終的に重み, $\alpha_{i,1}, \alpha_{i,2}, \ldots, \alpha_{i,n}$を得る.

$$\left[\alpha_{i,1}, \alpha_{i,2}, \ldots, \alpha_{i,n}\right] = Softmax(\tilde{\alpha}_{i,1}, \tilde{\alpha}_{i,2}, \ldots, \tilde{\alpha}_{i,n}) \quad (25)$$

以上では,i番目のトークンに対応する出力 α_iを計算する手法を述べたが, 異なるトークンに対する出力は別々に計算せずとも単一の行列演算で行うことが可能である . 下記のように, 入力, クエリ, キー, 値の出力のベクトルを縦に結合した行列を夫々 X, Q, K, V, Aとする.

$$X = vstack(x_1, x_2, \ldots, x_n) \quad \cdots\cdot (26)$$

$$Q = vstack(q_1, q_2, \ldots, q_n) \quad \cdots\cdot (27)$$

$$K = vstack(k_1, k_2, \ldots, k_n) \quad \cdots\cdot (28)$$

$$V = vstack(v_1, v_2, \ldots, v_n) \quad \cdots\cdot (29)$$

$$A = vstack(a_1, a_2, \ldots, a_n) \quad \cdots\cdot (30)$$

なお, $vstack$は, 行列要素を縦に結合する関数である . 出力Aは, Q, K, Vの関数として下記のように示される .

$$A = Attention(Q, K, V) = Softmax(\frac{QK^T}{\sqrt{d}})V \quad (31)$$

さらに, Q, K, V, は, 入力Xにより,

$$Q = XW^Q \quad \cdots\cdot (32)$$

$$K = XW^K \quad \cdots\cdot (33)$$

$$V = XW^V \quad \cdots\cdot (34)$$

以上はベクトル表現の式(20)〜式(22)の行列表現として位置づけられる.

5.4 Multi−Head Attention

具体的なTransformerでは, Scaled Dot−Product Attentionを拡張したMulti−Head Attentionが用いられている. Multi−Head Attentionでは, クエリ, キー, 値の組を複数用意し, 個々の組にScaled Dot−Product Attentionを適用し, 最後に一括出力する手法である .

ScaledDot−ProductAttentionでは,前層の出力x_iに対して,行列W^Q, W^K, W^Vを適用してクエリ q_i,キーk_i,値v_iを得た. そこで, 行列の組, $(W^Q_{(l)}, W^K_{(l)}, W^V_{(l)})$を複数用意しておけばクエリ , キー , 値の組を複数得ることができる$(l = 1, 2, \ldots, h)$. そのようにしてScaled Dot−Product Attentionの出力$a_i^{(l)}$が得られる. これらを連結してベクトルにし, 行列W_Oで線形変換することにより , 最終出力a_iが得られる.

$$a_i = hstack(a_i^{(1)}, a_i^{(2)}, \ldots, a_i^{(h)}) W_O$$

$hstack$は行列を横に結合する関数である . Scaled Dot−ProductAttentionの場合と同様に,Multi−HeadAttentionの場合も全てのトークンをまとめて行列演算として処理可能である. 個々の行列の組に対するScaled Dot−Product Attentionの出力を,全てのトークンについて縦に結合した行列を$A^{(l)}$とし, Multi−Head Attentionの出力を同様に処理して行列としたものを Aとすると,

$$A^{(l)} = vstack(a_1^{(l)}, a_2^{(l)}, \ldots, a_n^{(l)}) \quad \cdots\cdot (35)$$

$$A = vstack(a_1, a_2, \ldots, a_n) \quad \cdots\cdot (36)$$

他方, Multi−Head Attentionの出力Aは,

$$A = hstack(A^{(1)}, A^{(2)}, \ldots, A^{(h)}) W^O \quad \cdots\cdot (37)$$

夫々の行列の組に対するScaled Dot−Product Attentionの出力$A^{(l)}$は下記のように表される .

$$A^{(l)} = Attention(XW^Q_{(l)}, XW^K_{(l)}, XW^V_{(l)}) \quad \cdots\cdot (38)$$

5.5 Resudual Connection

Multi−HeadAttentionは,夫々のトークンに対して,ベクトルx_iをa_iに変換する$(i = 1, 2, \ldots, n)$. Transformer En-

coder では, Residual Connetionが用いられており, その際にはMulti−Head Attentionの出力が, 入力と出力の和として次の処理に送られる.

$$y_i = x_i + a_i \quad \cdots\cdots (39)$$

Residual Connetionにより, 深い階層を持つモデルに対しても学習が適切に可能になる.

5.6 Layer Normalization

Layer Normalizationは, 次の処理への入力を正規化するもので, 具体的には, ベクトルy_iが与えられた時に, その要素の平均値μ_iと, 標準偏差σ_iを用いて

$$LayerNorm(y_i) = \frac{\gamma}{\sigma_i} \odot (y_i - \mu_i) + \beta \quad \cdots\cdots (40)$$

を出力する. ここでβ, γはパラメータであり, y_iと同次元のベクトルである. $\odot$ は, 要素ごとに積を取る演算を表す.

5.7 Feedforward Network

Transformer Encoderの後半では, 先ず Feedforward Networkで処理される. 一つのLayer Normalizationが終了した後に, 夫々のトークンに対応するベクトルを z_iとするとここでの処理は

$$FFN(z_i) = GELU(z_i W_1 + b_1)W_2 + b_2 \qquad (41)$$

で表される. *GELU*関数は, 正規化線形関数（*ReLU*関数）を滑らかにしたような関数である.この後に,Resudual Connection とLayer Normalizationを適用したものが, TransformerEncoderの出力になる.TransformerEncoderの概念構成モデルを図8に示す.

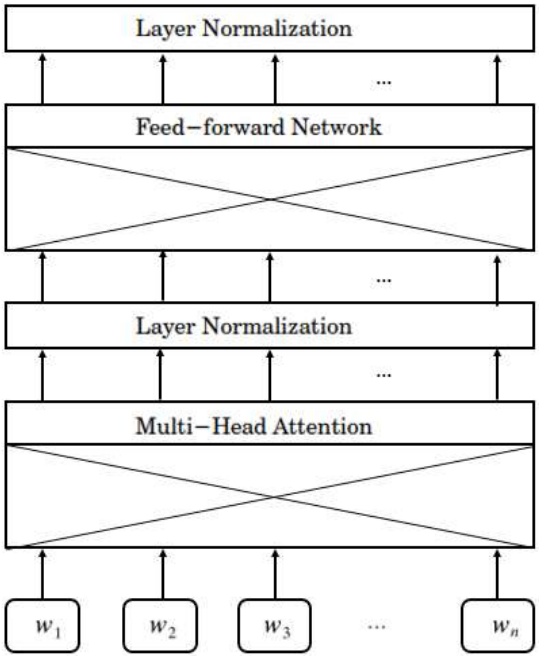

図8 Transformer Encoderの構成

MultiHead AttentionとFeedforward Network で構成され,その間にLayer Normalizationが組み込まれていることを示すが,図5の構成要素の枠組みと一致する.なお図5では, Layer NormalizationがAdd&Normと記されている.

6. 固有表現抽出

6.1 簡単な例

個々の語彙やトークンの属性管理の機構として固有表現抽出という技術がある.これは,人名や組織名のような固有名詞を抽出することを主眼とするが,それに関連して,日時のような時間情報, 位置情報を特定するための地名, 重量, 温度のような物理量,円やドルのような通貨単位,パーセントのような比率など広く使われる語彙の共通化を図ると共に, 語彙における意味的な属性を提供する.

例えば, BERTにおける例を紹介すると, 下記のような記述になる.

```
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
text = 'AさんはBCD株式会社を起業した.'
entities = [
  { 'name' : 'A' , 'span' : [0 , 1] , 'type' : '人名' , 'type_id' : 1}
  { 'name' : 'BCD株式会社' , 'span' : [4 , 11] , 'type' : '組織名' , '
    type_id' : 2} ]
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
```

ここで textは固有表現抽出を行う文章, entitiesはtextに含まれる固有表現のリストである. entitiesの要素は, 各固有表現に対応しており, name は名称の文字列, spanは文中での先頭からの位置で開始と終わりを示す. typeは型, type_idは型の分類識別である.

6.2 IO法による実装

固有表現抽出の実装法としては, IO法がある. これは, 各トークンにタグを付与することにより, 文章中の固有表現を抽出し, そのカテゴリーを示す. トークンを値とし, タグを要素名としてXMLで表現すると下記のようになる.

```
< IO法データ>
    < I : 人名> A </ I >
    < O > さん </ O >
    < O > は </ O >
    < I : 組織名 > BCD </ I >
    < I : 組織名 > 株式会社 </ I >
    < O > を </ O >
    < O > 起業 </ O >
    < O > し </ O >
    < O > た </ O >
    < O > . </ O >
</ IO法データ>
```

ここで, トークンが固有表現の一部であれば,そのタグは< I : type >となる. typeは固有表現のタイプを表す文字列で, ここでは人名や組織名を対象にしているが,先に述べた日時のような時間情報,位置情報を特定するための地名,重量, 温度のような物理量, 円やドルのような通貨単位, パーセントのような比率など, 想定される属性が対象となり得るのである. トークンが固有表現でなければ, タグは< O >となる. このようにして, 固有表現抽出技術に対してはXMLによるタグ付けが一つの手法となる.このようにXMLを使用して,オントロジ言語のOWLとの連携なども考えられる.

なお, 最近はXMLよりもJSONが使用される傾向が強いので, 関連データとの関連で実装も支配されると思われる.

6.3 従来技術との関係

固有表現抽出は，トークンとしての語彙や単語などの属性と値に関する動的なデータを活用する手法であり，その背後では，詳細な辞書を必要とする．というよりは，深層学習や利用者との対話を通じて，詳細な辞書を履歴として動的に作成するのが言語モデルであるという見方も可能であろう．この技術領域において，統計を用いない従来の論理的な人工知能やデータベース，マークアップ言語，オントロジ技術等との接点が存在すると思われる．

7. 考察

本稿執筆の動機は生成AI関連の議論の際に中枢の技術内容を把握していないことが気になったからである．この分野に関心を持ったのは，3年前の年次大会で「デジタル人文学とメタデータ」について検討した際にBERTに関する論文を読み，深層学習による自然言語処理が著しく進展していることを認識させられたことによる． 特にその中で，BERTとGPTとELMoを比較した図があり， その技術の詳細を知りたいと思ったことが背景にあった．その図は，本稿の図7，図6，図4そのものであるが，それを把握したいと思ったのである．

昨年の年次大会で，以上の経緯を思い出しつつ「デジタル人文学の可能性に関する考察～大規模言語モデルの登場による学際的分野の革新」[5]について発表した．この末尾で，若い人たちに生成AIに取り組むことを勧めながらそれを語る本人が生成AIの本質を把握していないことに後ろめたさを感じた．そこで一念発起して入手していた専門書[3]を具体的に紐解いたのであった．

本稿の執筆において，数式記述が重要な役割を占めているが，これは私が日本語化に関わったInterleaf5の後継システムであるInterleaf6のお陰である[6]．MicrosoftWordでも数式記述は可能であるが使い勝手は Interleafの方が優れているように感じる．このツールが無かったら本稿の執筆は到底不可能であった．書籍[3]は，統計数学的な観点ではていねいな記述がなされているので，式を追っていくことにより，統計的な概念が把握できた．

Word2vec, RNN, LSTM, ELMo, Transformer, 固有表現抽出の順で記述したが，Word2vecからTransformerまでは，数式が中心の記述となっている．図1～図8は，数式の理解を促進するために挿入したが，参照した専門書の図を参考に独自に描画した．この経緯から，専門書は読むよりはそれをトレースして記述しなおすことにより内容を把握できることを実感した．

本来であれば，プログラミングも併せて行うことが望ましいのであるが，Pythonによる記述は慣れていないのと環境にも不慣れなので諦めた．習熟しているLispであれば試みたかったが，環境の整備は困難であろう．DMH研究会の中堅ならびに若手研究者・技術者にはぜひPythonで生成AI分野のプログラミングに挑戦していただきたいと思う次第である．

DMH研究会で生成AIの中身について議論するような必要性は必ずしも無いと思われるが，博物館や美術館に関するコンテンツの生成や管理，統計的な評価などに関しては生成AI技術は今後重要になると思われる．特にコンテンツの分類，体系化などは，デジタルミュージアム，デジタル人文学において極めて重要である．そのためには，生成AIが内部でどのような数学的な処理を行っているかの基本的知識は必要と思われる．そのような目的のために，本稿がその端緒となることを意図して記述した．

8. おわりに

最近になって，ChatGPTに関して二度に渡って紹介する機会をいただいた．2023年10月30日に，NTT関係技術士会の皆様に，「ChatGPTと今後の世界」というテーマで講演した[7]．その翌月の11月17日に私の中学・高校の同窓生に対してほぼ同様の内容を講演したが[8]，いずれの場合も私と同年齢程度の高齢者を対象としたにも関わらず議論が非常に盛り上がった．ということから高齢者と言えども，生成AIに対する関心の高さを感じさせられたのであるが，残り少ない人生でも，今後の社会変化に対して前向きに生活することが重要なことを認識させられた．その経験をしたことからも，生成AIの基礎を数学的に把握することの意義を感じさせられた．

最後に 数式記述に関する優れた機能を提供してくれた米国Interleaf社のDTPシステムに感謝したい．このシステムは，Interleaf社CEOのDavid Boucher氏が構想し，Chief Technical Officerでプログラマであった Stephen Pelletier氏がコア部分を実装した．この二人とは30年前に親しく懇談・議論する機会を頂き，技術経営者並びに技術開発者としての生き方を強く印象付けられた経緯があった．ということからこの両名に感謝したい．

文献

[1] 大野邦夫, ”デジタル人文学とメタデータ”, 2021年度画像電子学会年次大会講演論文, Jun. 2021.

[2] Jacob Devlin, Ming−Wei Chang, Kenton Lee, Kristina Toutanova; “BERT: Pre−training of Deep Bidirectional Transformers for Language Understanding”, arXiv.org > cs > arXiv:1810.04805, Cornell University, 2018.

[3] 近江崇宏, 金田健太郎, 森長誠, 江間見亜利；“BERTによる自然言語処理入門”, ストックマーク株式会社編, オーム社, 2021.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, ?ukasz Kaiser, Illia Polosukhin, ”Attention Is All You Need”, 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, 2017.

[5] 大野邦夫,”デジタル人文学の可能性に関する考察～大規模言語モデルの登場による学際的分野の革新”,2023年度画像電子学会年次大会講演論文, Aug. 2023.

[6] 大野邦夫, ”オブジェクト指向雑記（その2） – プログラミングから分散オブジェクト・複合文書の世界へ– ”, 画像電子学会誌, Vol.52, No.3, pp.429−434 , 2023

[7] 大野邦夫,“ChatGPTと今後の世界”,NTT技術士会講演資料, Oct. 2023.

[8] 大野邦夫, “ChatGPTと今後の世界～振り返らずに前を向いて生きよう”, 栄光学園12期生講演会資料, Nov. 2023.