FERI INFORMATION

エグゼクティブのための生成 AI 講座

その３　生成 AI が文脈を理解するしくみ

副島　豊（SBI 金融経済研究所・研究主幹）

文章とは単語が並んだもの

シリーズその１では、1980 年央に登場した AI 第二世代の代表例であるニューラルネットワークが LLM（大規模言語モデル）の基本モデルになっていることを紹介した。ニューラルネットワークのみならず、コンピュータプログラムは、インプットデータやアウトプットデータとして数字を扱う。このため、自然言語をコンピュータで処理するためには言葉を数字化する工夫が必要となる。シリーズその２では、単語を数値化するベクトル化技術（分散表現技術）について概観した。

ところで、文章は単語の並びから成り立っており、その順番が文意や文脈を作り出している。数値化された単語をニューラルネットワークで扱う場合にも、単語の並びが重要となる。

しかし、当初開発されたニューラルネットワークは、インプットデータに順序性はなかった。順番をどう並べても、インプット値に対応するパラメータが変わるだけであり、並びの順番情報は活用されていなかった。

株価のような時系列データは並びが決定的に重要であり、順序を入れ替えると重要な情報を失ってしまう。ニューラルネットにおいて時系列データや文章のような順序性に意味があるデータを扱うには新たなモデルが必要であった。

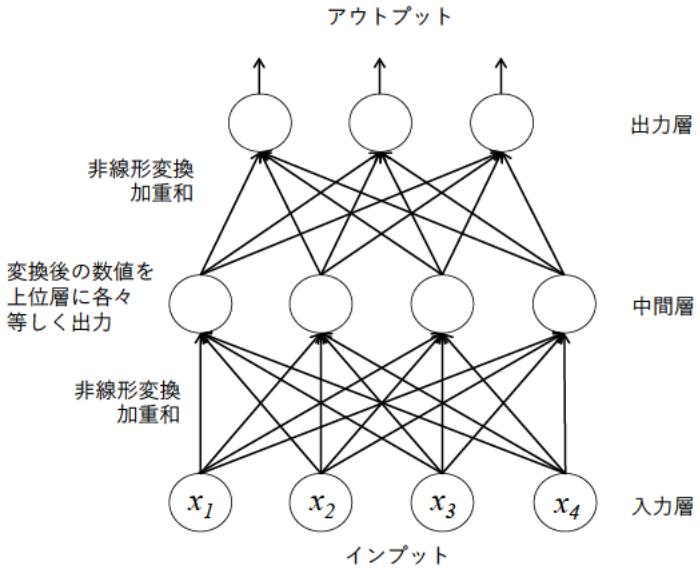
1990 年に登場した RNN（Recurrent Neural Network）では、データの順序情報をモデル内で扱えるような工夫がなされた。RNN は時系列データを扱えるモデルとして応用が進んだが、この時期はニューラルネットワークの学習能力の限界や問題点が判明した時期と重なり、その後、いわゆる AI 冬の時代を迎えることになった。このため RNN モデルが注目される機会は減っていった。

ところが、自然言語処理（Natural Language Processing）の研究が進展するなかで、2000 年代初頭にニューラルネットを自然言語処理に応用することで文章を生成するモデル（言語モデル）を作る試みが始まった。そして、2010 年には RNN を言語モデルに応用した研究が登場し、RNN は再び注目を集めることになった。

入出力の順序性

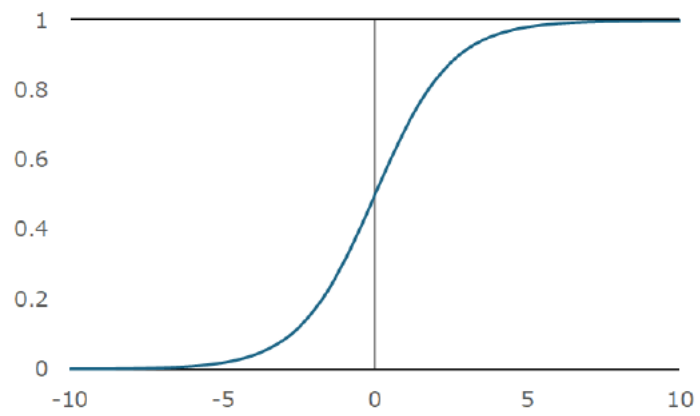
ニューラルネットワークの基本形は、次図に示したように各インプットの線形和（一定の比率をかけて足し合わせたもの）を非線形変換し、これを中間層の各要素とする。中間層は複数あり、各インプットは異なる比率で足し合わされ、各中間層に代入される。こうして作られた中間層の値を更に線形和・非線形変換してアウトプットデータとして出力する。

中間層が１層のニューラルネット



非線形変換の代表例はシグモイド関数であり、正負の様々な値を０～１の間の値に変換する関数である（次図）。これは、非線形性の導入により関数全体の表現力を高める役割を持つ。また、入力層から各中間層に来る加重和の値が大小様々に異なっていても、中間層の値を０～１の間に基準化する役割も有している。

シグモイド関数



さて、インプットデータが時系列情報である場合、x_1から右側に向かって、x_2, x_3, x_4と順に並ぶ。ここで、x_1と x_2を入れ替えたとしても、中間層に向かって集計されていく際の加重ウエイトのパラメータが入れ替わるだけであり、順序が変化したことの影響を捉える手段がモデル側にない。

図示したようなニューラルネットの基本形では、入力層から中間層、出力層と一方向に情報が加工・集約されていくため、フィードフォワード・ニューラルネットワークと呼ばれる。初期のニューラル言語モデルはこれを利用していた。

単語の分散表現のアイディアは 2000 年に登場し、初期のニューラル言語モデルでは、単語をワンホットベクトルで表現し（その 2 の脚注 2 を参照)、これをインプットとしたうえでベクトル表現化はニューラルネットの内部で計算していた。前回解説した単語の分散表現の代表例である word2vec は NN をより単純化した線形変換であり、NN の最下層に一体化して取り扱うことも可能である。前処理として単語のベクトル表現化を事前に行うようになったのは 2013 年に登場した word2vec 以降である。NN 本体に取り込んで一体表現したほうが非線形変換が入る分、表現力は高まる。しかし、実際は計算負荷や学習の難しさから、ベクトル化表現は別のモデルで行い、これを NN のインプットとするアプローチが後の LLM でも一般的となる。また、いったんベクトル化表現モデルを作ると様々な LLM に適用可能という汎用性がある点も、こうしたアプローチを可能とした。この背景には、ある問題領域（ドメイン）で学習したモデルを別の問題領域に転用することができるという転移学習能力を NN が有している点が関係している。

状態を記憶する RNN

当初のニューラル言語モデルでは、 t 番目の単語をアウトプットとして生成するために、直前の n 個の単語をインプットデータとしていた。これは、単語や文脈の繋がりを反映させるために必要な処置であった。しかし、計算負荷の観点から扱える語彙数を絞る必要があり、また n 個の単語の前後関係情報も活用されなかった。

これに対し、RNN では過去にどのようなインプットがあったかという情報を累積・更新しながら引き継いでいく変数が導入された。これを隠れ状態ベクトル h_tと呼ぶ。次図のように、一期前の隠れ状態ベクトル h_{t-1}と今期のインプットベクトル x_tを合算することで、隠れ状態をインプットごとに更新していく。

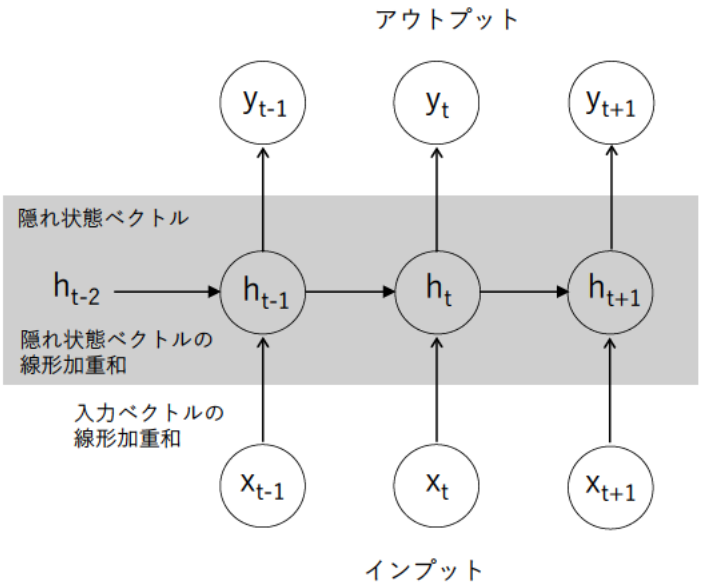
この図では各インプットや一期前の隠れ状態から出ている矢印が一本で表現されているが、言葉を分散表現化することで一つの言葉が高次元ベクトルに変換されているため、実際には多くの数値を加重和として集約している。隠れ状態も高次元ベクトル

であるため、こちらも加重和したうえで、インプット側と合算される。これにバイアス項（定数項）を加えたうえで非線形変換し、アウトプット層へ引き渡される（これがどう言語生成に繋がるのかは後述）。

なお、線形加重和や非線形変換のパラメータは、各 t 期において共通しており、インプット内容に応じて変化したりはしない。言い換えると、どのようなインプットや隠れ状態ベクトルに対しても共通して対応する高い汎用性を持つ。

隠れ状態ベクトルは、過去のインプットを順次集約して積み重ねていったものであるため、言葉の並びが作り出す意味・文脈情報が埋め込まれている。こうした RNN の仕組みにより、文章に対応する能力が高まった。

RNN の基本モデル



RNN もディープ NN 化が可能

図では、隠れ状態ベクトルは 1 階層となっているが、複数の隠れ状態ベクトルを階層構造として導入することができる。これは、ディープニューラルネットワークのアプローチと同様である。すなわち、基本形のフィードフォワードニューラルネットワークでは、中間層を複数階層に積み上げる（ディープにする）ことで、インプットとアウトプットの間にある複雑な情報構造を捉える能力を高めることができる。言い換えると、インプットをアウトプットに変換する関数として、より複雑で高性能なものを作り出すことができる。これがディープラーニングモデルの特徴となっている。

RNN の隠れ層にも同じコンセプトを適用し、深く階層化することで能力向上を図ることができる。

数学的な解析により、ディープラーニングモデルが高い性能を持つ理由が明らかになっている。まず、中間層のうち低層部分において、インプット情報から特徴量（データに内在するパターン性）を中間層の各要素別に集約する。上層部では、得られた様々な特徴量の組み合わせによってアウトプットを表現している。こうした仕組みにより関数としての性能（近似精度）を高めている。ただし、こうした原理が数学的に解析されているのは、モデルが複雑でなく極端にディープでないものに限られており、それ以外のものは上手く機能する理由が未だに判っておらず、研究が続いている。

ディープラーニングモデルは膨大なパラメータを持つ。その推計法の基本的な考え方は 1980 年代央に考案された AI 第二世代モデルから変わっていない。教師データ（正解を示したアウトプットデータ）にフィットするよう膨大なパラメータ群を誤差逆伝播法により推計する。このパラメータ推計のことを学習と呼んでいる。これは、正解を示す教師データにアウトプットが近づくようニューラルネットを学習させるという考え方から由来している。その際に、アウトプットの誤差から中間層のパラメータを、次にインプット層のパラメータを逆方向に修正していくため、誤差逆伝播法と呼ばれる。誤差を最小化する過程では段階的にパラメータ修正を繰り返す方法がとられており、微分の考え方を用いた再急降下法という手法が適用される。

ニューラル言語モデルは確率モデル

RNN が言語を生成する仕組みを理解するために、まず、一般的なニューラル言語モデルの言語生成原理から説明する。LLM の問題点としてハルシネーション（幻覚、もっともらしい嘘）が指摘されているが、確率モデルとしてのニューラル言語モデルの成り立ちを理解すると、いわゆるハルシネーションの発生は必至であることが判る。こうした LLM の原理を理解しておくと実務への応用時に使い道を誤るという失敗が回避しやすくなる。さらには、GPT のような汎用 LLM がどのような限界を抱え、その課題をどのように乗り越えているかを知り、実装に活かしていくかを学ぶ上でも有益である（今後のシリーズで紹介予定）。

言語モデルで文章を生成する場合、以下のような手順が一般的である。ある単語の並び、例えば、{日本/の/首都/は/} があったとする。次にくる単語を予想すると、「東京」が最有力候補であろう。もちろん、「暑い」でもよいし、「京都/だった」が続くこともありうる。しかし、蓋然性が最も高いのは東京であろう。

確率言語モデルも同様な発想に基づく。条件付確率 P(東京|{日本/の/首都/は/})を求めて、東京以外の他の全ての単語候補と比較し、一番確率が高いものを選択して、これまでの文章に付け足す。これを次々に繰り返すことで文章が生成されていく。

広辞苑の収録数は約 7 万語である。それより遥かに多くの単語について、確率計算を行うのは膨大な計算負荷が生じる。このため、様々な計算節約手法が考案されているが、文章生成の原理は上述の通りである。最も高い条件確率を示した単語を選択するか、多少のゆらぎをいれるかなどバリエーションが選択・程度調整可能な仕組みが取り入れられている。

こうした LLM の言語生成の仕組みは、その用途ほどには知られていない。2022 年に ChatGPT の登場により LLM に社会的な注目が集まったが、当時、LLM を検索サービスのように誤用することが広範に生じた。LLM は学習用の膨大なコーパスに存在している単語間の登場のパターン性を学習し、次の単語を確率的に繋ぎあわせることで文章を生成しているにすぎない。にもかかわらず、整理された知識が LLM から引き出せるかのように利用され、結果、ハルシネーションを起こす点が問題視された。

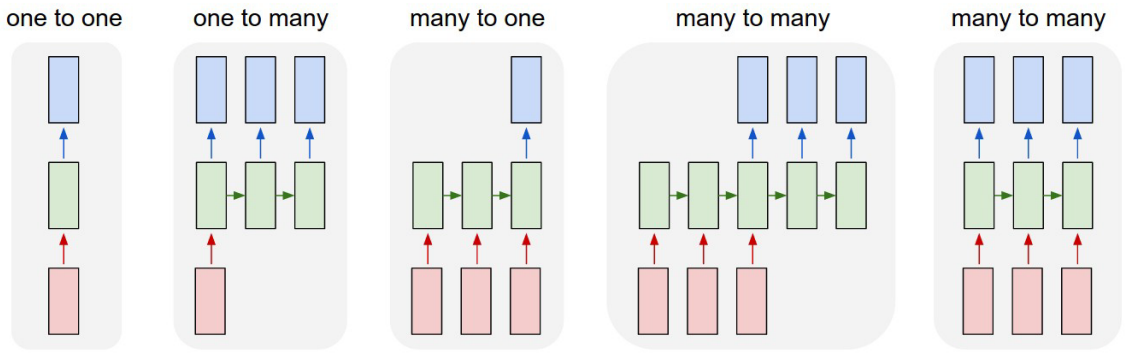
その後、LLM が高度化・巨大化し、学習のためのデータセットも膨大なものとなったため、あたかも LLM が膨大な知識を保有しているかのように見える度合いが格段に高まった。もちろん、単語間の関連性や登場順に関するパターン性が極めて精緻に把握されれば、LLM に知識が蓄えられたと捉えることもできる。このため、上述のような誤解が解消されにくい状態が続いており、これが LLM の用途の適切な選択に悪影響を及ぼしている面がある。

RNN の応用バリエーション

前出の RNN の解説図では、インプットとアウトプットが一対一に対応していたが、異なる設計も可能である。代表的な設計例を次図に示した。一番左は、インプットとアウトプットが 1 対 1 に対応している事例である。最もシンプルなこのケースでは、中間層があるものの、その内部の隠れ状態ベクトルが横方向（時間展開方向）に引き継がれていないため、RNN とはいえない。他事例の説明のための基本形として取り上げたものである。

しかし、このシンプルなケースも用途は多々あり、例えば、画像認識や分類・識別問題がこれに相当する。１つの画像をインプットとし、それが何を示しているかをアウトプットとした画像の分類・識別の事例である。

RNN の様々な設計例



出典）Andrej Karpathy, “The Unreasonable Effectiveness of Recurrent Neural Networks,” May 2015.

左から 2 番目は、画像からの文章生成事例である。文章を単語の逐次生成によって作っていくため、アウトプットが順次作成されている。一つの画像インプットから最初の単語を生成し、隠れ状態ベクトルを更新しながら、その単語が生成されたもとでの次の単語を選択していく手続きをとる。

3 番目は、文章からの内容分類などが相当する。アウトプットは内容に関する複数の候補であり、逐次入力されるインプットを最後まで受けたうえで、アウトプットが生成される。例えば、この文書のトピックは何か（国際政治、経済、スポーツ、芸能、金融市場等）を識別させる問題である。企業の決算報告を読み込んだうえで、今後の株価が、上がる・下がる・横ばいという離散選択肢を選ばせる問題もこれに相当する。

4 番目は翻訳が典型例である。日本語の文章をインプットとし、最後まで読み込んだところで、英語の文章を順次生成していくケースに相当する。5 番目は、同時通訳やビデオ画像へのキャプション文書生成などに相当する。4 番目との違いは、インプットの読み込みを最後まで待たずに、順次アウトプットを生成していく点である。

seq2seq：エンコーダー・デコーダー

上記のように RNN の応用モデル形が広がるなか、2014 年に RNN を 2 つセットで利用する seq2seq モデルが提案され、機械翻訳などへの応用が加速した（正確には RNN の発展形である LSTM<後述>を利用している）。

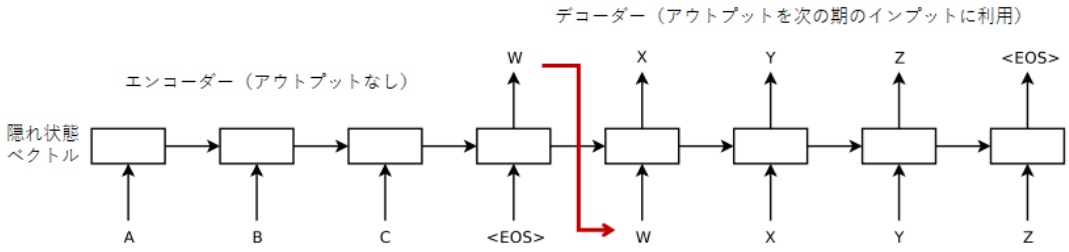
seq2seq では、一つの RNN でインプットとアウトプットを対応させるのではなく、前半の RNN でインプット情報を隠れ状態ベクトルに集約させ、これを後半の RNN の隠れ状態ベクトルの初期値として引き渡し、アウトプットを順次生成していく

手法である。次図のように、後半の RNN のインプットには、一期前のアウトプットが用いられている。日英翻訳の場合、日本語文章を読み取った後、生成された英単語が逐次的に翌期のインプットとなる。

単語の並びとしての文章内容を高次元の固定ベクトル（ここでは隠れ状態ベクトル h_t）に変換することを、エンコーディング（符号化）と呼び、これを順次文章に復元することをデコーディング（復号化）と呼ぶ。2 つの RNN をエンコーディングとデコーディングに特化させて使うアプローチであり、のちに LLM の性能を劇的に改善させた Transformer モデルも、エンコーダー・デコーダーモデルとして作成されている。

エンコーダー部分はアウトプットを持たず、デコーダー部分のアウトプットを翻訳結果とすることで 2 つの RNN がセットで学習されることになる。通常の RNN では各期の線形加重和や非線形変換のパラメータは共通となっているが、seq2seq ではエンコーダー部とデコーダー部の RNN でパラメータが異なる分、モデルの自由度が高まり、学習性能も向上する。

Seq2seq モデル



出典）Ilya Sutskever et.al, “Sequence to Sequence Learning with Neural Networks,” 2014.

注）EOS は文章の終了を示す。オリジナル論文の図表に筆者が加筆。

なお、seq2seq モデルの用途は機械翻訳に限らない。文章を与えて要約文を返す、質問を与えて回答を返す、対話を与えて対話を返す、文章を与えて文法誤り修正後の文章を返す、口語文章を与えてビジネス文章に変換して返すなど、文字系列を異なる文字系列に変換する様々な用途がある。

seq 2 seq モデルは、言葉の意味からすると、文章のような系列インプットを系列アウトプットに変換するモデルを指すことになる。しかし、エンコーダー・デコーダーという仕組みを取り込んだ研究によって seq 2 seq という名称が定着したため、エンコーダー・デコーダーモデルを含意することが一般的である。ほぼ同時期に同様なモデルを開発した研究論文は、seq 2 seq ではなくエンコーダー・デコーダーモデルという呼称を用いている。

RNN の 2 つの課題

こうして登場から 10 数年の時をおいて、時系列データから自然言語処理に用途が拡大した RNN だが、モデルの構造上、 2 つの課題を抱えていた。一つは、隠れ状態ベクトルのアップデイトをどれほど強く行うか、すなわち、新しいインプットの情報を強く反映させるか、それとも過去のインプット情報を長く保有させるかというトレードオフである。

前者の場合、過去の情報が損なわれやすく、文脈を長く保持しておくことができない。逆に後者の場合、新しい単語がもたらす文章内容の展開について行けなくなるという問題がある。適切な単語の選択は直前の単語の影響を強く受けるため、直前や近い位置のインプット情報を重視せざるをえない。このため、RNN では遠く離れた単語や文脈をうけて次の単語を適切に選択することが難しかった。言い換えると、隠れ状態ベクトルにおいて長期記憶を保持することが困難であった。

もう一つの問題は、学習に誤差逆伝播法（の応用系）を適用する際に、計算が不安定化もしくは進行しなくなり精度が上がらなくなるという学習時の難点であった。

RNN が自然言語処理に適用される以前から、こうした問題は認知されており、解決する方法が考案されてきた。1990 年代央から後半にかけて登場・改善されてきた LSTM（Long Short Term Memory）が代表例である。2010 年代には RNN と同様に自然言語処理への適用が進展した。その後、LSTM の構造をよりシンプルにし計算・学習効率を高めた GRU（Gated Recurrent Unit）も考案されている。その 4 では、LSTM や GRU の仕組みについて解説する。

なお、RNN や LSTM は隠れ状態が入れ子上になって時間展開しており、誤差逆伝播法では計算負荷が大きくなる問題も別途、存在していた。これは、Transformer モデルに組み込まれた Attention 機構で解決された。同モデルは生成 AI の性能を劇的に改善させた点が最大の特徴であり、これにより BERT や GPT などに代表される LLM 時代が 2010 年代末に到来する。大規模言語モデル（LLM）という言葉は、Transformer モデルの登場とそのモデル規模拡大による著しい性能拡大によって登場し、普及していくことになる。

（その 4 につづく）