生成AIウォークスルー：基本技術、LLM、アプリケーション実装

副島 豊｜SBI 金融経済研究所研究主幹　兼　SBI ホールディングス
SBI 生成 AI 室プロジェクトコーディネーター

要約

本稿は、生成 AI とりわけ大規模言語モデル（LLM）の発展と実装技術に関する展望論文である。ニューラル言語モデルの基礎、LLM に繋がる理論モデルの発展、様々な LLM の群雄割拠、発展過程で発見されてきた多様な転移学習の形態、それがもたらした利用法の拡大、企業での活用法（秘匿情報を扱う手法）、実装に必要な技術群とその学び方、最後に、実際の構築事例を紹介している。生成 AI 技術の活用には手を動かして学ぶことが重要であり、6 つの学習ステップをその一例として示している。LLM の仕組みやシステム実装に関する本稿の情報は、生成 AI 技術を金融実務に活用していく際の学びや実践の水先案内（Pilot）となろう。

副島 豊

SBI 金融経済研究所研究主幹　兼
SBI ホールディングス SBI 生成 AI 室プロジェクトコーディネーター

1966 年生まれ。京都大学卒、90 年日本銀行入行。フィンテックセンター長や金融研究所長を歴任。90 年代より様々な先進的分析手法を日本銀行に導入。金融システムレポートや各種レポートを企画・創刊。BIS・グローバル中央銀行活動のエキスパートメンバーとして国際基準策定等に参画。

1．はじめに

ChatGPT が彗星のように登場し、人々の耳目を集めたのは 2022 年末であった。僅か 1 年半前のことである。これまで自然言語処理（NLP: Natural Language Processing）や AI 技術とは縁がなかった人々が、文章でコンピュータに指示を出し、その都度コンピュータによって生成された文章で回答を得るという新しい体験を享受した。コンピュータが文章や画像、音声などの情報を直接処理し、オリジナルなものを新たに生成するという新技術の普及は、ビジネスや日常生活、社会活動に無限の可能性をもたらす。多くの企業や個人、公的セクターが生成 AI 技術の有用性を強く意識するようになり、生成 AI は 2023 年に最も注目された言葉の一つとなった。金融ビジネスにおいても、対顧客サービスや企業内利用において様々な活用法が検討され、実際にサービスとして提供され始めている。

一方で、生成 AI を検索サービスのように利用する誤用法や、これに伴うハルシネーション（幻覚、もっともらしい嘘）が話題となった。また、生成 AI の学習時における利用情報や、そのプライバシー保護、情報セキュリティ管理、生成内容の公正性や倫理面での問題[1] など、様々な課題も意識されるようになった。

1：宇根（2022）は、金融サービスの提供において機械学習による予測・推論を利用する際の公平性・公正性に関する諸問題を俯瞰している。同研究の対象は機械学習であるが、生成 AI にとっても同様な問題が存在している。

生成 AI サービスの創造と活用の促進、および上述のような課題への対応においては、①生成 AI がどのような技術であるか、その中身を知り、②どのような IT インフラの上に構築されているか、構築していけばよいのかを知り、③急速に進化を遂げる生成 AI 技術のフロンティアをキャッチアップし続けること、が必要となる。①の理解が不足していると、生成 AI の利用が適切でないタスクを選択したり、機械学習など他の手法が適切な分野に生成 AI を応用してしまい、望ましい成果を得られない危険性がある。②の理解が不足していると、アイディアはあっても実用化に至らないか、陳腐化が著しい（寿命が短い）生成 AI 活用サービスにおいて外注を繰り返すことになる。ビジネスモデルの DX（デジタルトランスフォーメーション）を巡る議論では、デジタル社会におけるサービス構築や業務構築には IT 知識が必須であり、内製化やビジネス推進の現場が IT を学ぶ必要性がより強く意識されるようになってきている。生成 AI も同様であり、現場が技術を理解していないと、優れたサービスを発想し実装することは難しい。また、③の対応をとらないと、より高度な生成 AI 技術やそのサービス実装方法が次々に誕生し、機能・コスト・実装の容易さが劇的な速度で改善していくもとで、競争優位性があるサービスを提供することが困難となる。本稿は、金融ビジネスに携わる企業やビジネスパースンが、上記①～③にかかわる基礎的な知識を習得できるよう、生成 AI 技術の中核となっている LLM（大規模言語モデル）の原理と発展、実装方法をできるだけ平易に俯瞰することを試みている。

学術研究においても生成 AI の活用が生産性や競争力を高めるうえで重要となっている。知的創造活動は生成 AI の恩恵を最も享受しやすい分野である。その高い情報集積・処理能力は、論文執筆における文献レビュー、研究対話や議論などの壁打ち、仮説生成、データ分析、分析プログラム作成、論文構成、文書校正などに有益である。例えば、文献レビューを効率化する生成 AI アプリケーションサービスとして Google 提供の notebookLM がある[2]。同サービスは、RAG と呼ばれる生成 AI の活用法（本論文で解説）を極めて簡便に利用できるようにした生成 AI アプリケーションサービスである。PDF や Web ページで提供される情報をまとめてデータベース化し、この情報に基づいた質疑や節ごとの要約、理解度チェックなどが利用できる。本年 6 月より日本語対応版が提供され始めており、Google 開発の LLM 最新モデル Gemini 1.5 Pro がバックエンドで稼働している。6 月末現在で Experimental 版として無償提供されている。

図表 1 は、筆者の最近の論文、副島（2024）を PDF アップロードして情報データベース化した画面である。論文の概要が示されており、目次タブをクリックすると各節の要約が LLM によって作成され、簡便に文献レビューを行うことができる。下方には、論文に関する質問を受け付ける窓がある。回答は論文情報の範囲内で作成されるため、ハルシネーションが生じにくくなっている。作成根拠となった論文中の箇所が示されるため、回答内容の確認や論文中の当該箇所へのアクセスも行いやすい。学習ガイドのタブでは用語集や、内容理解度を確認できるクイズ（解答付き）などが提供される。その一部を図表 2 に示した。

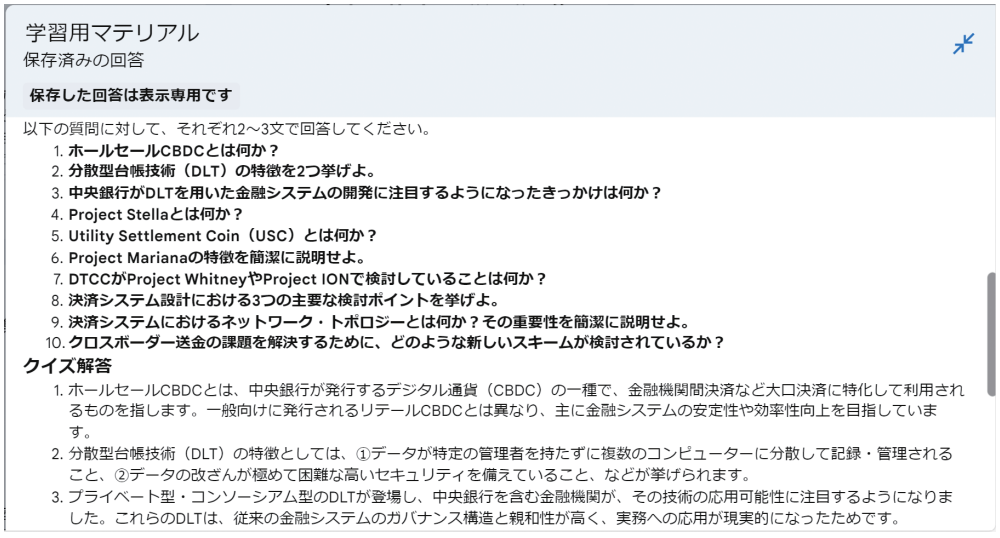
2：ChatGPT は、その機能をカスタム化して特定のサービス構築を可能とするプラットホーム GPTs を提供している。同サービス上では、Consensus や Scholar GPT、Paper interpreter など文献検索やレビューに特化した便利なサービスが提供されている。ChatGPT の有償ユーザになると各種の GPTs を使用できる。GPTs サービスの作成者は自作サービスが大量利用された場合は、これを収益化することができる。

図表３は、SBI 金融経済研究所の Web サイトで直近１年間に公表された Web レポート 24 本を html アドレス指定で読み込ませて情報データベースを作成した画面である。ある研究分野の代表的な論文を大量にデータベース化すると、同分野に精通した生成 AI サービスを作成することができる。こうした有益なサービスが次々に誕生しており、生成 AI の開発動向や用途の発展を知ることは研究者にとっても重要となっている。

図表１　notebookLMの画面：副島（2024）論文の読み込み

出所）筆者作成

図表２　学習ガイドで自動作成されたマテリアルの一部：理解度クイズ



出所）筆者作成

図表3　SBI金融経済研究所Webレポートの情報データベース化

出所）筆者作成

本稿の構成は以下のとおりである。２節では LLM の基礎となったディープニューラルネットワーク（DNN）型の自然言語モデルについて解説する。3節では、自然言語モデルの発展について解説する。特に、Transformer モデルの登場によって急速な性能向上がもたらされ、その後に LLM 時代が到来したことや、これが現在の最新 LLM にどう展開していったかについて解説する。４節では、LLM 発展の過程でスケール則や創発現象、転移学習が発見され、これらが LLM 開発の方向や応用手法を決めていったことを紹介する。5節では、非公開情報を利用する生成 AI サービスの主要な開発手法となっている RAG（Retrieval Augmented Generation）について、追加学習を行う Fine-tuning と対比させながら解説を行う。６節では、生成 AI アプリケーションを実装する方法について、IT インフラや実装技術の進化の面から解説する。専門家でないと携わることができなかったアプリケーションのシステム実装において、参入障壁を引き下げる民主化の波が押し寄せていることを簡単な RAG の構築事例を示しながら紹介する。最後に７節では、LLM の学際的研究の可能性として、人間の認知や行動の理解という視点から、World Models や人工市場シミュレーション、経済理論モデルへのインプリケーションを議論する。

2．生成AIの中核技術LLMを支える基礎技術

自然言語処理の研究は長い歴史を持つ。言語をコンピュータで解析し理解する技術（自然言語理解、NLU）として、言語の文法や規則性を研究しルール化することでテキストを解析する手法などが 1950 年代から考案されてきた。文章を単語の固まりに分解する形態素解析や、固有名詞などの固有表現抽出、文章のツリー構造を判別する構文解析、述語項構造認識[3]、コーパス[4] や類義語辞書などの作成などである。テキスト分析やテキストマイニングも NLU の一分野であり、コーパスなどから自動的に情報を抽出・分析し、知識を整理・発見したり、仮説を検証する研究などが行われてきた。文書の分類（トピック分類や性質分類 < 例えばスパム / 非スパムメール >）、ポジティブ・ネガティブ等の感情分析、書き手の判別（紫式部の文章か否か）などである。言語の生成アルゴリズムを対象とした自然言語生成（NLG）の研究により、機械翻訳や文書要約、対話の創造などの手法が発展してきた。コンピュータが一般化した 1980 年代以降は NLP に統計的手法が導入され、大規模なコーパスを用いた単語の登場頻度や順位、単語間の関連性（隣接して出現する共起発生の頻度 <n-gram> など）の研究が行われた。

ニューラルネットを応用したニューラル言語モデルの登場（Bengio et al. 2003）によって自然言語処理へのアプローチが変わり始めた。そして、2017 年に登場した Transformer モデルにより、言語理解や生成のパフォーマンス（例えば機械翻訳の精度）の面で、過去の長い研究蓄積を短期間のうちに凌駕していった。本節では、まず、ニューラル言語モデルが確率モデルとして文章を生成していく方法を解説し、次に、ニューラル言語モデルの基礎となっているニューラルネットワークについて説明する。最後に、言語をコンピュータで処理できるよう数値化する手法、特に膨大なコーパスを活用して単語間の関係性を抽出した数値化を行う手法（単語の分散表現 / 埋め込み表現）を解説する。

2.1 ニューラル言語モデルの確率的文章生成

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。条件付き確率 P（東京 | {日本 / の / 首都 / は /}）を計算し、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、これまでの文章にその一単語を付け足す。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。固有名詞を含めると日本語にはそれより遥かに多くの単語が存在する。例えば、日本語 Wikipedia をコーパスとして単語数を計測した事例を見ると 130 ～ 150 万の範囲で報告されている（web クロールのコーパスを含めると 200 万）。これらのすべてについて上記のような確率計算を行うと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り、最も高い条件付き

3：誰がいつどこで何をしたという術語に関わる構造を文章から取り出す（認識する）こと。

4：自然言語として書かれた文章や使い方を大規模に収集し、コンピュータで検索できるよう整理されたものをコーパスと呼ぶ。文章を構造化したうえで品詞など言語的な情報を注釈として付与したものが一般的であり、文章をそのまま集めたものは生コーパス（raw corpus）と呼ばれる。文書の多様性を考慮して構築された均衡コーパス（balanced corpus）、複数言語の翻訳をセットにした対訳コーパスなど、様々なコーパスが作成されている。かつては紙で作成されていたが、現代ではコンピュータで扱えるようデジタル情報化されている。これにより統計処理、例えば、ある文書における特定単語の登場頻度に注目した TF-IDF 分析などが容易に行えるようになった。詳細は岡崎他（2022）を参照。

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のバリエーションが選択・調整可能な仕組みが取り入れられている[5]。

こうした言語生成の仕組みは、その用途ほどには知られていない。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選択し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、整理された正しい知識や情報が 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。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、ハルシネーションの発生は必至であることがわかる。

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。一方で、単語間の関連性や登場順、組み合わせなどに関するパターン性が様々なトピックについて極めて精緻に把握され、その結果、正確な情報を提供する能力が高まってきており、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くなっており、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。

しかし、ここ 1 ～ 2 年のモデル規模や学習情報の巨大化により、知識データベースとしての有用性が加速的に高まっている。例えば、MMLU（Massive Multitask Language Understanding）という言語モデルの汎用性、理解力、推論能力を評価するための包括的なベンチマークテストがある。多言語かつ様々な領域の知識理解を一般常識から専門知識まで幅広く試すものである[6]。その評価スコアによると、最先端の LLM は一般人を遥かに上回る専門家並みの知識を有しているという指摘がなされている[7]。同テストは選択形式であり、回答文書生成時のハルシネーション発生リスクは別途存在しているが、言語生成における汎用能力も改善を続けており、公開情報の範囲内での知識に基づく言語生成に限定すればハルシネーションのリスクは低下しつつある。また、ハルシネーションを回避する技術的な工夫もなされている。例えば、RAG を活用して根拠となるオリジナル情報の該当箇所を提示したり、プロンプト技術（入力する指示情報に詳細な条件を付加することで意図した結果を引き出す工夫）によってハルシネーションを予防する工夫などである。このほか、特定分野の専門情報を追加学習させることによって知識を強化することで一般情報に基づく誤答の可能性を回避するような LLM も開発されているため、知識データベース的な活用が更に進むと予想される。本稿の執筆においても、最新の LLM 群や LLM 活用サービスを情報検索などに活用している。

2.2 ニューラルネットワークの仕組み

LLM の基盤となっているモデルはニューラルネットワーク（以下、NN）である。1980 年代前半に登場し、第二次 AI ブームを牽引した NN は、脳のシナプスの構造を単純化して表現したモデルである。図表 4 のようにインプッ

5：例えば、①確率が一番高いものを逐一選ぶ（greedy decoding）、②複数個先までのセットで確率を評価する（beam search）、③確率的ゆらぎをいれる（random sampling）といった方法がある。③では、上位 p%（あるいは k 個）の候補からランダムに選択する方法や、temperature という手法が知られている（確率のばらつきの差を人為的に縮小させることでゆらぎが起こりやすくするもので、GPT が temperature や p% 法を採用している）。①は分類問題、②は機械翻訳、③は長文生成でしばしば利用されている。

6：各種の LLM リーダーボードでも評価基準として活用されている。3 節では日本語性能を評価している LLM リーダーボードを紹介している。

7：Hendrycks et al.(2021)、OpenAI et al.(2023) を参照。

トとアウトプットの間を中間層によって繋ぎ、アウトプットが正解（教師データ）と合致するようパラメータを調整する仕組みである[8]。下層の要素を線形和し、これを非線形変換したものが上層の要素となる。これはシナプスが他の多数のシナプスからの電位差を受け取り、その合計値がある閾値に近づくと急速に（非線形的に）出力が上昇して発火する（新たな電位差を作り、隣接するシナプスに伝える）仕組みを模倣したものである。

線形和をとる際の係数や定数項（バイアス）、非線形変換のパラメータを変えることで、図表４中の４つのインプットは中間層の各要素（ノード）に異なる値をもたらす。中間層からアウトプットに向けても同じ仕組みが適用される。インプット変数の組み合わせ比率を変えることで中間層の各ノードに異なった情報を集約し、インプットとアウトプットの間にある何らかの関係性をノードが分担しながら捉えようとするものである。これは、機械学習における特徴量抽出と類似した機能である。こうしたNNは、推論時にはインプットからアウトプットに向かって前向きに情報が伝わっていくため、フィードフォワード型NNと呼ばれる。NNの最も基本的なモデルとなっている。

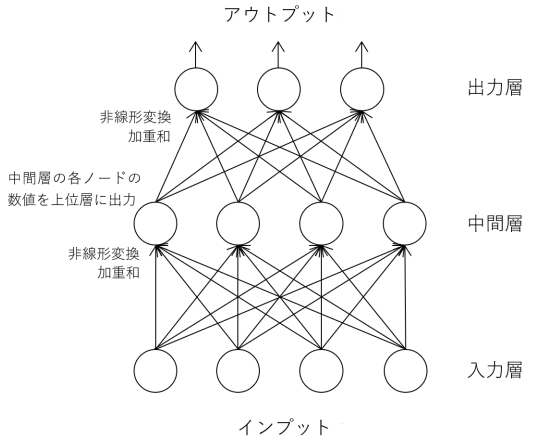
非線形関数には、図表５で示したようなシグモイド関数、ハイパーボリック・タンジェント関数、ReLU（Rectified Linear Unit）やその変形判が用いられる[9]。シグモイド関数の場合、アウトプットが１つであれば０か１という判別問題に対応でき、選択肢が３つであれば、アウトプットを３つ（y_1, y_2, y_3）とし、教師データを（1,0,0）、（0,1,0）、（0,0,1）とすることで判別問題に対応できる。なお、1要素だけが1という値をとり、他要素がゼロであるベクトルを、1要素だけが発火しているという意味でone-hotベクトルと呼び、ニューラル言語モデルでも重要な役割を担っている（2.3節の分散表現で後述）。

このようにインプットとアウトプットを繋ぐのは四則演算やシンプルな非線形関数に過ぎない。しかし、各要素は単純な関数であっても、インプット数や中間層の要素数を増やし、中間層を複数積み重ねる（図表４では１層のみ）ことで、複雑に機能する非線形関数を作りだすことができる（今泉 2021）。

8：この推計をNNでは学習と呼ぶ。学習には、アウトプットと教師データの誤差最小化問題が利用され、微分による漸近的なパラメータ調整を、アウトプット層から中間層、インプット層に向かって進めるため、誤差逆伝播法（バックプロパゲーション）と呼ばれる。

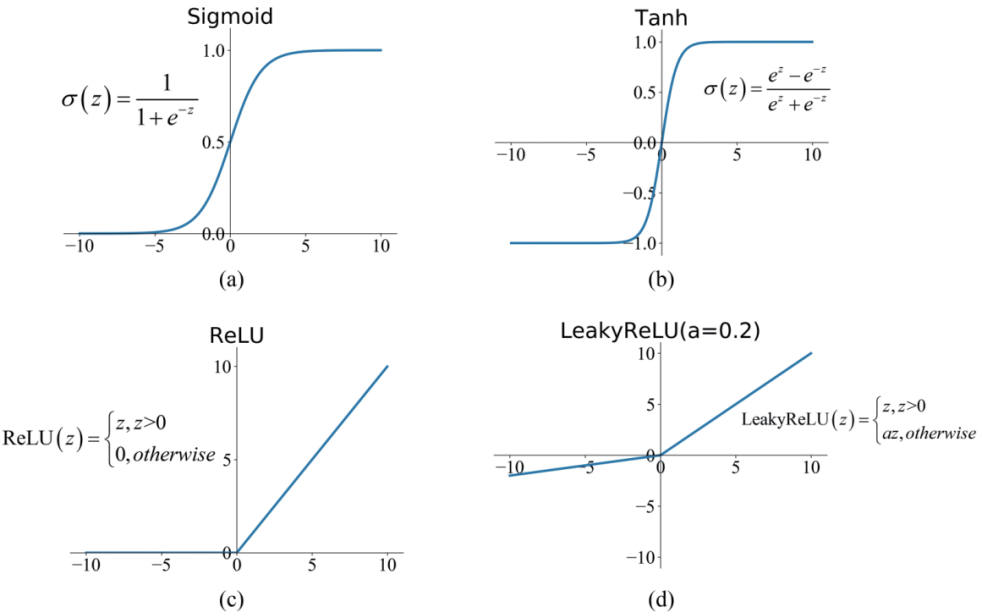
9：シグモイド関数は０～１内の値をとるため加重和の標準化が行えるほか、学習の際の誤差逆伝播法で微分値が簡便な形となるメリットがある。ReLUはシグモイド関数の弱点である勾配喪失問題（微分値がゼロに近づくと学習が進まなくなる問題）を回避するのに効果的である。

図表4　ニューラルネットワークの基本構成



出所）筆者作成

図表5　非線形変換の事例



出所）Feng et al.（2019）

金融分野の応用問題としては、デフォルト判別・与信判別分析や格付け推計モデルがある。こうした判別・分類問題に対しては、計量経済学の手法として質的選択モデル / 離散選択モデルが知られており、このほか機械学習の分野でも様々な手法が考案されてきたが、NN は関数の表現力が非常に高く、これが優れた判別能力をもたらした[10]。副島（1996）は、金融政策反応関数（当時は公定歩合の上げ / 下げ / 据え置き）に単純な構成の NN を適用し、政策変更を高い精度でインサンプル推計している。論文中では、NN 関数の柔軟で複雑な非線形性を可視化している。著名な金融政策反応関数として線形のテイラールールがあるが、これより遥かに複雑な関数形状を示しており、かつ、同じインフレ率、実質成長率に対しても、これらが加速している場合と減速している場合では、政策反応関数の形状が異なることを示している[11]。

NN は中間層の階層やインプット数を増やすとモデルの精度や表現力が高まることが理論的にも期待された。しかし、中間層を積み増して深くする（ディープにする）と学習が進まなくなるという技術的問題に直面し、80 年代の第二次 AI ブームは徐々に収束して、その後、長い AI 冬の時代が訪れた。ところが、2000 年代後半にこの限界をブレイクする技術が現れた。ディープニューラルネットワーク（DNN）であり、これが第三次 AI ブームを切り開いた。中間層を数十も積み重ねてもパラメータの学習が上手く行われる手法の開発や計算能力の確保によって、NN のパフォーマンスが飛躍的な向上を辿り始めた。LLM には NN/DNN を発展させたモデルが利用されており、これらを 3 節で解説する。

10：シグモイド関数型 NN の中間層を無くすと、質的選択モデルの代表例であるロジットモデルと一致する。シグモイド関数はロジット関数と呼称名が異なるだけであり関数形は同一である。質的選択モデルではパラメータ推計に最尤推定法が用いられるが、NN では確率的勾配降下法が誤差逆伝播法において利用される。岡崎他（2022）の第二章が両者を比較しつつ詳細に解説している。

11：副島 (1996) の図表 17、18 を参照。また、インフレ率や実質成長率の加速度が同じでも、これらの水準が異なると政策反応は当然異なってくる。

2.3 言葉の数値化：分散表現/埋め込み表現

ニューラル言語モデルの進化とあわせてLLM時代をもたらした要素に、言葉を数値化する手法、いわゆるベクトル化技術（分散表現や埋め込み表現[12]とも呼ばれる）がある。その代表例が2013年に登場したword2vecである。

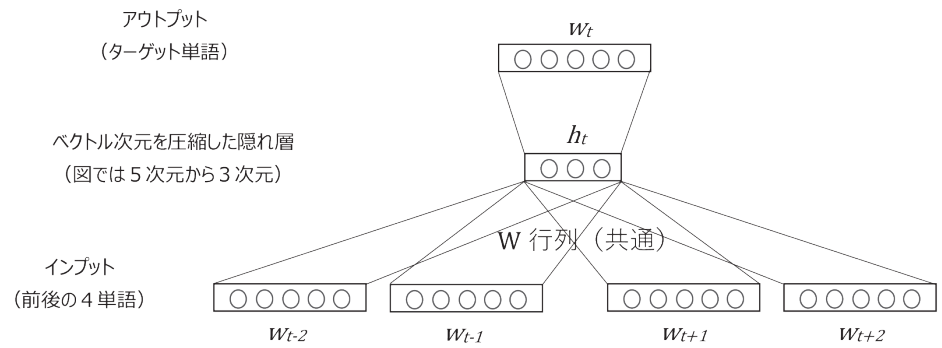
ニューラルネットワークのインプットやアウトプットは数値である。それゆえニューラル言語モデルでは、言葉を数値化する必要がある。単純なアプローチとして単語にIDを振る方法が考えられるが、膨大な量になるだけでなく、文章に内在する単語間の関係性をどのように効率的に表現するかという問題に直面する。仮に単語が100万語あって、単語1と残りの全単語との関係を個々に数字で表現していくと、その情報量は100万×100万の行列となり非常に効率が悪い。当初は、実際の文章に出てくる単語の「離接」関係（何単語分離れているか）を用いて単語をベクトル表現化するアプローチが採られた。

新しいアプローチをもたらしたのはword2vecというモデルである（Mikolov et al. 2013）。これは、文章中の前後の単語（複数単語でも可）から、間にある単語を推測する、あるいは、ある単語から前後の複数の単語を推測するという推論問題をNNに学習させるものである。前者をCBOW（Continuous Bag-of-Words）、後者をskip-gramと呼ぶ。穴埋め問題そのものに意味があるわけではなく、推定されたパラメータ（図表6におけるインプット層と中間層を結ぶ加重パラメータW）が単語間の関係性を凝縮した行列となっており、その各列が各単語の分散表現となるよう穴埋め問題が設計されている。図表6のモデルは非線形変換を含まないためNNモデルではないが、線形関係だけで表現されたモデルのため、当該単語と前後の数単語の関係をコンパクトな次元空間に押し込む/埋め込む（embed）ことができる。これが分散表現や埋め込み表現と言われる所以である。

CBOWを示した図表6・7に沿って説明する。インプットは前後の単語をone-hotベクトル表現したもの4つである。単語総数をN（例えば100万）とし、押し込む次元をhとすると、重みWは$N \times h$の行列となる。入力値が4つあるため、h次元のベクトル4つが中間層に向かって出力され、その平均値をとることで中間層の$\tilde{h}$ベクトルを作成し、これに$h \times N$のW'行列を乗じてN次元ベクトルに戻し、当該単語のone-hotベクトルを教師データとして、これにフィットするようWやW'を学習する（ただし、分散表現獲得のために必要としているのはW行列のみである）。ここではt番目の単語w_tを穴埋め問題のターゲットとしているが、このターゲットを文の冒頭近くから終点近くまで変えていくことで、学習データセットの組み合わせを多く用意できる。かつ、1文だけでなく膨大な文を対象に同様なデータセットを準備すると、どの問題にも共通して適用される固定パラメータWは、単語間の平均的な関係性を表現することになる。

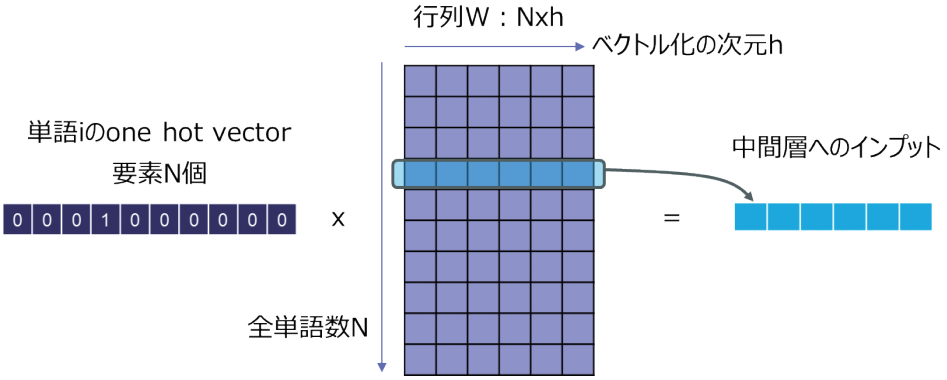
12：ある単語や文章を、高次元ベクトル空間、例えば1,536次元ベクトルに押し込んで表現するため、Embedding（埋め込み表現）と呼ばれる。Embeddingは、次元削減（高次元のデータを低次元に圧縮）や、特徴量エンジニアリング（データ特性を数値ベクトルに変換）にも用いられる。

図表6　word2vecのモデル構造



出所）筆者作成

図表7　ウエイト行列が分散表現となる仕組み



出所）筆者作成

単語をベクトル化することで以下のような言葉の計算が可能となる。

東京－日本 ＝ パリ－フランス

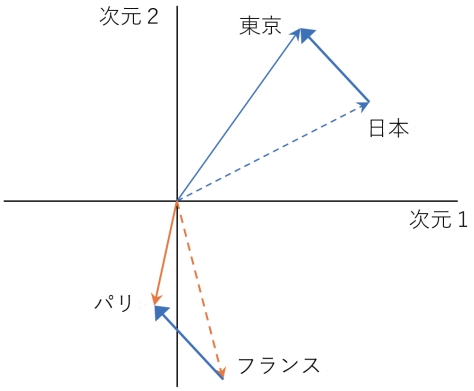
これは、概念上の情報処理ではなく、数字（ベクトル）の計算として成立しうるものである。各単語が２次元空間に埋め込まれてベクトル化されているとしよう（h=2 のケース）。このとき、あらゆる単語がこの平面上のベクトルとして表現される。日本やパリという単語が図表８に示したベクトルとして表現されたとしよう。上式左辺は図表８の太い青矢印ベクトルであり、これは右辺（オレンジで示した２つのベクトルの差）と同一となっている。青矢印は、ある国の国名と首都という単語間の関係性を示すベクトル表現に相当する。学習対象のコーパスに「日本の首都は東京である」「フランスの首都はパリである」といった単語の関係性を示す文章が数多く存在し、その関係性を word2vec が集約するため、４つの単語の関係性を分散表現で表すことが可能となる。

ベクトル化により単語の類似性の計測も図表９のように可能となる。単語が数値化されているのでコサイン類似度や空間内での近傍関係を測る指標を活

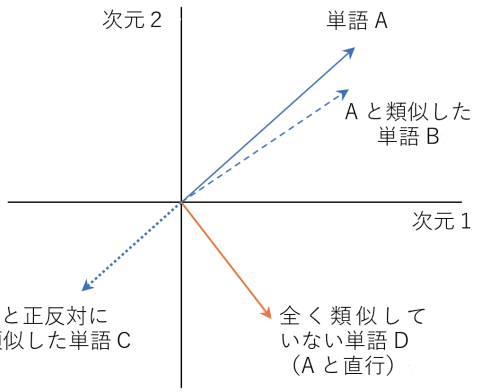
用することができる。EC や映像音楽の配信などで各種の推薦サービスが提供されているが、こうした推薦サービスには画像や商品特性を特徴量抽出し、デジタル情報化（ベクトル化）したうえで、類似度を計算するという技術が活用されている。

コサイン類似度はベクトルの内積表現から得られている。内積は 2 つのベクトルが成す角が 0 度に近い（同じ方向を向いている）ほど大きくなり、直行するとゼロとなる。内積が類似度や関連性の高さを示すことは、後述の Transformer モデルの Attention 機構で活躍するだけでなく、ニューラル言語モデル全般において重要な役割を果たしている。

図表8　パリ=フランス+（東京−日本）



図表9　二次元空間で測った単語の類似性



出所）筆者作成

次元数 h は様々な設定が可能であるが、単語間の関係性を表現するためには数百～数千次元のものが使われることが多い。N 個（仮に 100 万個）の単語間の関係を示すことに 100 万× 100 万の行列を使うことは不効率である。N が 100 万だとしても、埋め込み表現を使うことにより次元空間 100 万× h まで圧縮することが可能となる。単語の分散表現は、単に言葉を数値化するだけでなく、こうした次元圧縮によって LLM の性能向上や計算コスト削減をもたらしている。機械学習の特徴量表現においても膨大なデータ内にある関係性を同様な次元圧縮技術を用いて抽出しており、コンセプトは同じである。なお、ある分散表現モデルで埋め込み次元数 h を決定すると、どのような単語であっても固定長 h 次元のベクトルに変換される[13]。

こうして word2vec は注目を集めることとなったが、学習対象のコーパスに含まれる単語数が 100 万のように大きなものになると計算量は膨大なものとなる。Negative sampling のような効率化手法が考案されたが限界があった。また、単語の隣接関係に基づく推論問題から算出されるため、位置関係より更に複雑な文脈の流れや、複数の意味や読み方を持つ文脈依存の多義語を扱うのが不得意であった。なによりも、word2vec は言語を生成するモデルではなく、単語を分散表現する手法であり、生成 AI の実用化はその後の LLM の発展があってのことであった。

なお、分散表現のモデルは言語生成モデルと並行して発展を続けており、有名なものとしては FastText（Bojanowski et al. 2016）や ELMo（Peters

13：英語のように単語間にスペースで区切りがある、すなわち分かち書き（tokenization）がなされている言語の場合、分散表現によって各単語がベクトル化される。一方、日本語のように分かち書きしない言語では、その言語専用に開発された形態素解析で品詞に分解する前処理作業を行うことが伝統的な自然言語処理において一般的である。こうしたトークナイゼーションを行うツール（トークナイザー）としては MeCab や ChaSen、JUMAN、Janome などが知られている。しかし、分散表現モデルを適用すると、形態素解析とはかなり異なる切り方、人間には解釈が困難な切り方をする場合が少なくない。言葉のパターン性や単語・文字レベルでの関係性の把握が、人間の認知とは異なる方法で行われていると考えられる。本稿では単純化のために、単語単位にトークン化されるという想定で説明を行っている。モデル解説で単語と呼称している箇所のなかには正確にはトークンと呼ぶべきケースがあるが、わかりやすさの観点から単語と表記している。

et al. 2018）がある[14]。ELMo（Embeddings from Language Models）では双方向 LSTM（LLM の前駆的モデルの一つ、後述）を用いて、文章全体の情報に基づき単語を分散表現し（contextualized word embedding）、かつ１つの単語に３つのベクトル（各々 1024 次元）をもたせることで多義語対応などのパフォーマンスを改善している。word2vec のように前後の数単語との関連性情報だけでなく、LLM を使うことによって文脈を反映させた分散表現となっている点が大きな改善点である[15]。

ChatGPT で知られる OpenAI もベクトル表現に特化した LLM を開発している。例えば、GPT-3 や 4 では、text-embedding-ada-002 という分散表現に特化した軽量高速のモデルが用いられている[16]。ただし、GPT など近年主流となっている LLM の基礎となった Transformer モデル（Vaswani et al. 2017）では、分散表現の過程を分離するのではなく LLM の内部に取り込む手法が登場している。初期の Transformer モデルである BERT（Devlin et al. 2018）がこうした流れを作った[17]。この点は 3.4 節で後述する。にもかかわらず、分散表現モデルの開発が続いているのは、RAG でベクトル化データベースを作成するニーズ（後述）や、運用の効率性追求、計算資源の節約といったが理由が存在しているためである。

３．LLM時代をもたらしたニューラル言語モデル

LLM の発展には、単語が意味ある順序で並んでいるという文章の系列データ的特性を表現できるモデルが必要であった。株価の時系列データを並び替えると意味がなくなるのと同様、単語も文書内でランダムに並び替えることはできない。その意味で、文書は単語の時系列データである。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、既存の RNN モデル（Recurrent NN）を自然言語処理に応用しようという試みが登場した。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状態変数ベクトルが導入されている。また、同モデルの欠点を補う LSTM モデル（Long Short Term Memory）も開発されていたので、RNN と並行して LSTM の応用も進められた。しかし、両モデルとも状態変数を導入したことにより、NN のパラメータ算出に用いるバックプロパゲーションが関数の入れ子構造をとることになった。これが計算負荷や学習における難点となっていた。また、インプットとして長い文章を扱うと、入れ子構造が長大化するため、長文を処理することが苦手であった。この問題を解決したのが Transformer モデルの Attention 機構である。３節では、これらのモデルを順に解説する。なお、本節の内容は、各モデルを提唱した原論文に加え、岡崎他（2022）、Rothman（2021）、坪井他（2017）、斎藤（2018）を参考にしている。

14：これらのほかにも、word2vec と同様な発想で文章を分散表現するモデル Doc2Vec が考案された（Le and Mikolov 2014）。しかし、Transformer モデルで LLM の内部に分散表現を取り込んでいく動きが主流になったことや、対象が文章全体であったことから（単語単位、文単位、数文単位など区切り方を自在にできる分散表現のほうが便利である）、発展はしなかった。

15：文中の単語をマスクし予測させることに加え、文の前後関係の正誤判別トレーニングという２つの学習手法を適用することで、文脈を捉える能力を高めている。単語レベルでの分散表現獲得では、word2vec は前後の単語の的確な選択訓練に skip-gram を用いているが、これを文単位版で学習する発想に類似している（正確には、文そのものを予測はせず、順番をバラバラにした候補文＜教師データ＞の中から適したものを選択するという学習である）。

16：2023 年末には、text-embedding-3-small/large にモデルチェンジしている。

17：これに対し、ELMo は双方向 LSTM という LLM を使いつつも分散表現を作成することが主眼となっており、それ自体に文章生成能力はない。BERT は LLM の機能のなかで分散表現を行っており、主たる用途はエンコーダー機能（文書分類等）である。

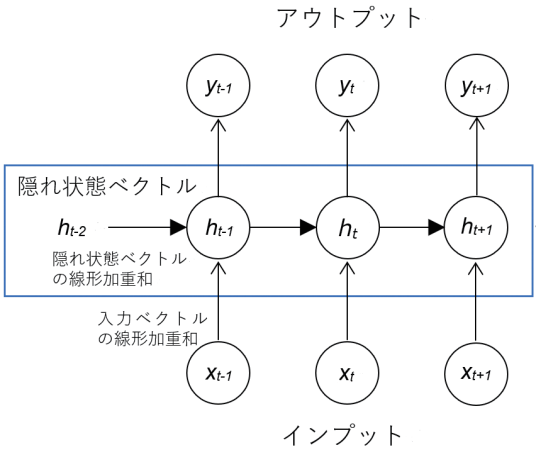
3.1 RNN：系列データへの対応

1980 年代央に登場した初期のフィードフォワード型 NN はインプットの順序性を考慮しなかった。どのような順番で並べても、対応する重みパラメータ群が入れ替わることで対応できたが、見方を変えると、株価のような順序性がある時系列データや、文章における単語のような系列データにおいて、並び方が持つ情報を捉えることができなかった。この問題に対応するため、NN が登場して間もないころから、時間展開や前後展開の情報を取り扱える RNN モデルが Jordan（1986）や Elman（1990）によって提案されていた。

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。図表 10 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。Recurrent は再帰という意味だが、隠れ層の情報がアウトプットに向かうだけでなく、内部（翌期の隠れ層）に再帰するため、こうした用語法が用いられる[18]。なお、時系列モデルの視点からは、RNN は状態空間モデル（カルマンフィルター）の NN 版と捉えることができる。

18：図表 4 で示したフィードフォワード型の NN とは別に、Hopfield（1982）は全要素が結合したホップフィールドネットワークを提唱している。これは、出力がネットワーク内に再帰（recurrent）するという点で RNN の前駆的なモデルとなっている。

図表10　RNNの基本モデル



出所）筆者作成

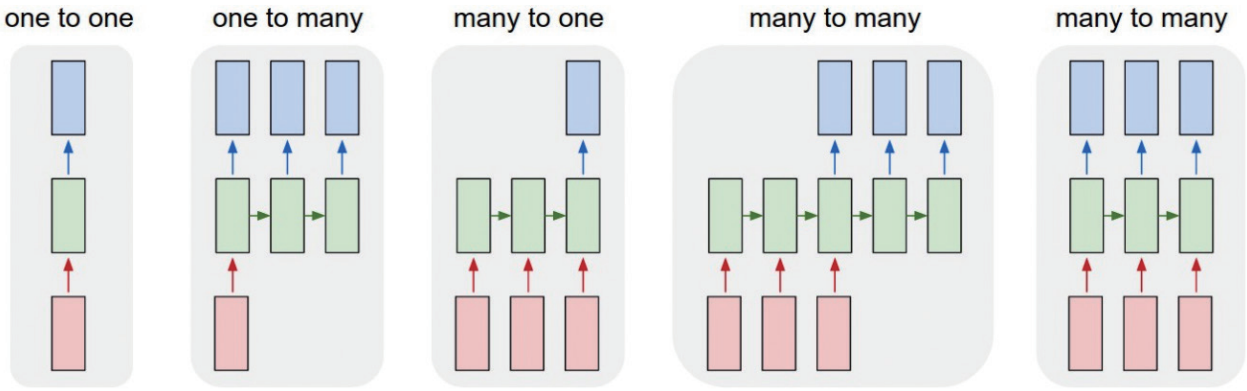
RNN は時系列データを扱えるモデルとして応用が進んだが、この時期はニューラルネットワークの学習能力の限界や問題点が判明した時期と重なり、その後、いわゆる AI 冬の時代を迎えることになった。このため RNN モデルが注目される機会は減っていった。ところが、自然言語処理の研究が進展するなかで、2000 年代初頭にニューラルネットを自然言語処理に応用することで文章を生成するモデル（ニューラル言語モデル）を作る試みが Bengio et al.（2000）や Bengio et al.（2003）により始まった。そして、Mikolov et al.（2010）によって RNN を言語モデルに応用した研究が登場し、RNN は再び注目を集めることになった。

図表 10 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、分散表現によって各単語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。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。図表 10 では、隠れ状態ベクトルは 1 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。図表 4 に示した基本形のフィードフォワード NN 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。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可能である。

次に、RNN には様々な応用バリエーションがあることを紹介する。図表 10 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、様々なバリエーションが設計可能である。代表的な設計例を図表 11 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。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。

図表11　RNNの様々な設計例



出典）Karpathy (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分類候補であり、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。

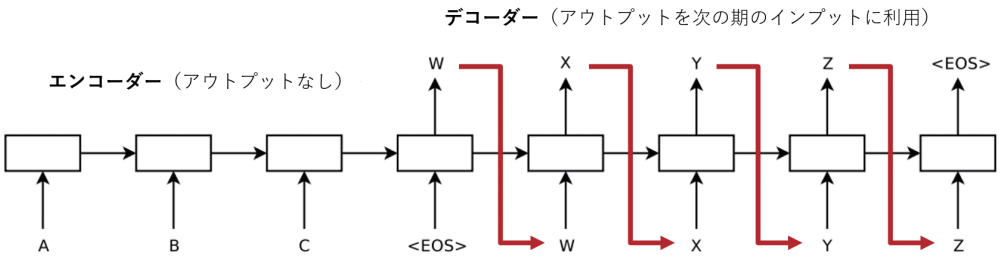
3.2 エンコーダー・デコーダーモデル、seq2seqモデル

上記のようにRNNの応用モデルが広がるなか、RNNを２つセットで利用するseq2seqモデルがSutskever, Vinyals and Le（2014）によって提案され、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を利用している）。図表12に示したように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。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。例えば日英翻訳の場合、日本語文章を読み取った後、生成された英単語１つが逐次的に翌期のインプットとなり、隠れ状態ベクトルと合わせて翌期のアウトプット（次の英単語）を生成する。隠れ状態ベクトルには前半で読み取った日本語

の情報が蓄積されており、英単語を一つ生成するたびにアップデイトされていく。

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、デコーダー部分にアウトプットを生成させることで 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。

図表12　seq2seqモデル



出典）Sutskever et al.（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。

seq2seq モデルは、その名前の意味からすると、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。ほぼ同時期に同様なモデルを開発した Cho et al.（2014a）は、seq2seq 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。3.4 節で示すように Transformer はその名前の通り、機能的には系列変換モデルであり、モデル構造としてはエンコーダー・デコーダーモデルを採用している。Transformer 系統のモデルが主流となった近年においては、汎用アーキテクチャとしてのモデル構造に注目してエンコーダー・デコーダーモデルという呼称が使われることが多い。

こうして登場から十数年の時をおいて、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問題である。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。

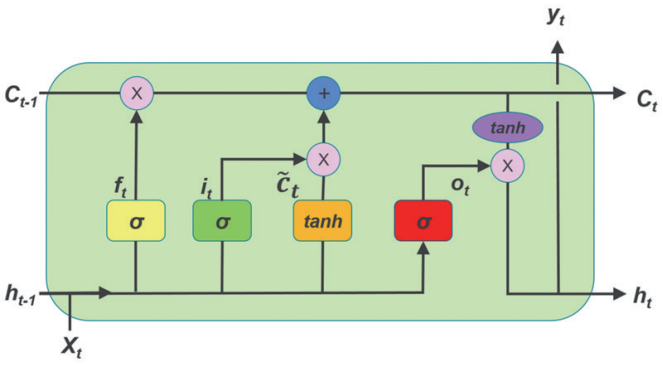
RNN が自然言語処理に適用される前から、こうした問題は認知されており、解決する方法が考案されてきた。1990 年代後半に登場してきた LSTM（Long Short Term Memory）が代表例である。2010 年代になると RNN と同様に自然言語処理への LSTM 適用が進展した。

なお、RNN が抱える 2 つの課題のうちのもう一つは、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。この問題は LSTM でも解決できておらず、Transformer モデルの登場を待つことになる。

3.3 LSTM：短期記憶と長期記憶

Hochreiter and Schmidhuber（1997）は、前述のような RNN の課題について長期記憶と短期記憶に役割分担させることで解決を図った。図表 13 は隠れ層の内部を示している。h_t は RNN と同様、隠れ状態ベクトルを示すが、その役割は短期記憶に特化している。長期記憶は c_t が担っており、左上の前期値から引き継がれてきた c_{t-1} は、まず忘却関数 f_t を通過する。忘却の程度の調整は、短期記憶 h_{t-1} と今期のインプット x_t の線形和にシグモイド関数 σ を適用することで0～1の範囲で調整される。この仕組みは忘却ゲートと呼ばれている。次に新規情報を長期記憶に追加する。新規情報も h_{t-1} と x_t から作成され、別の加重和とハイパーボリック・タンジェント（tanh）関数によって作成されたベクトル情報（図中ではセルゲートを通過する $\tilde{c}_t$、g_t と表記する文献もある）に、入力ゲートの重み（入力ゲート関数 i_t で作成された0～1内の値）を乗じたものを加算する。この重みも h_{t-1} と x_t から作成される。これを翌期に引き渡す（図中の右上に抜けていく c_t）ほか、短期記憶 h_t は長期記憶 c_t の一部を残すことによって作成される。まず、c_t に tanh 関数をかけて基準化し、これに出力ゲート o_t からの0～1調整を乗じて、今期の短期記憶 h_t とするとともに、これをアウトプット y_t として利用する。出力ゲート o_t も h_{t-1} と x_t から作成され、シグモイド関数によって範囲調整される。

図表13　LSTMの短期記憶と長期記憶のアップデイト



出所）Zheng, Yuan and Chen（2017）

RNN では中間層に相当する h_t は更に加重和と非線形変換を通じてアウトプットに変換されていたが、LSTM では短期記憶をアップデイトしたものをそのままアウトプットとしている。今期の短期記憶は、今期の長期記憶を tanh 関数で基準化したものが元情報になっており、２つの状態変数ベクトルがアウトプットに近い情報セットとなっている。見方を変えると、長期記憶の更新が RNN における隠れ層からアウトプット層への変換に相当していると解釈することもできる。こうした設計は LSTM の計算効率を高めるうえでの工夫として導入されている。

LSTM のモデルの特徴の一つは、あらゆる情報が短期記憶 h_{t-1} の加重和と今期インプット x_t の加重和の和で更新されていく点である。忘却ゲート、入力ゲート、出力ゲートを作成するパラメータ群は全て学習によって算出され、これらの違いが各種ゲートの機能の相違や長期記憶に付加する内容そのものを生み出している。Transformer モデルの Attention 機構（QKV 機構、後述）では、こうした発想が更に大規模に活用される。

RNN が自然言語処理に適用され始めたのは 2010 年であったが、1997 年に登場していた LSTM も同時期に自然言語処理に応用され始める。前述したように 2014 年には seq2seq モデルに LSTM を適用した研究が登場している（Sutskever, Vinyals and Le 2014）。なお、SLTM と類似したモデルとして GRU（ゲート付き再帰ユニット）がある。本稿では説明を省略するが、そのエッセンスは、長期記憶は導入せずに状態変数ベクトル h_t についてリセットゲートや更新ゲートを通じて更新していくというものである。GRU は最初から自然言語への応用を念頭に開発されたものである（Cho et al. 2014b）。Cho らは、同年にエンコーダー・デコーダーモデルの出発点となった研究も行っている（前述の Cho et al. 2014a）。

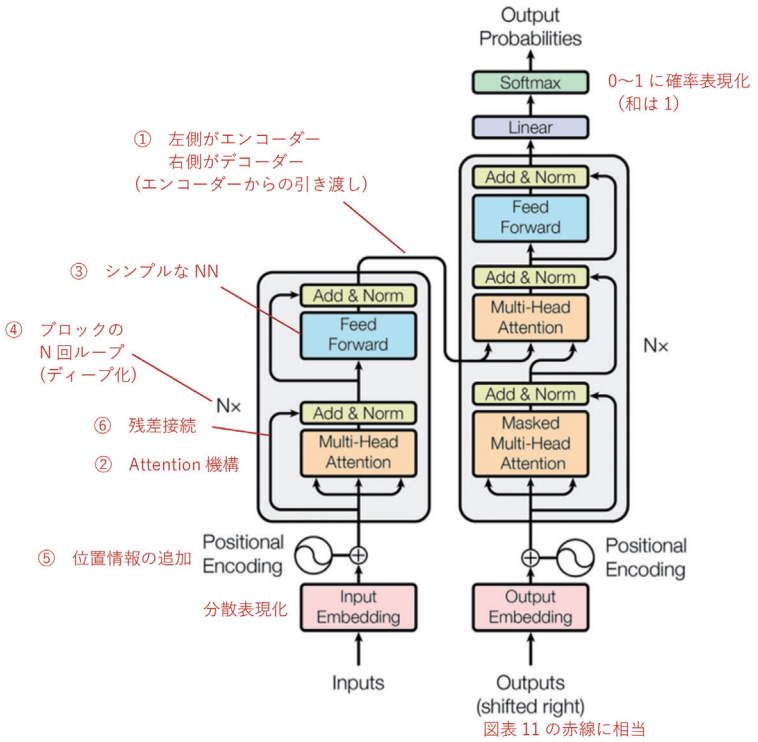
３.４ Transformerモデル：Attention機構によるゲームチェンジ

RNN、LSTM、GRU モデルは、いずれも再帰モデルであった。モデルが時間方向・系列方向に入れ子構造となっている点は情報更新の面で機能的であるが、学習の際の計算負荷が高く、かつ学習が進みにくい（勾配消失問題や勾配爆発問題）。加えて、学習後の推論（インプットを与えてアウトプットを得

る）の際にも逐次的処理となり、言語モデルとして利用する際に、並列処理による時間短縮ができないという問題があった。入れ子構造は長文処理能力の強化においても制約となり、長文処理の制約によって複雑な文脈構造や指示関係も捉えにくくなるという限界もあった。

これらの点を改善したのが、Attention 機構を持った Transformer モデルである。“Attention is all you need”というタイトルが付けられた Vaswani et al.（2017）は、現在の LLM 大進化の起点となった。同モデルは、図表 14 で示しているように、①エンコーダー・デコーダーモデルの適用、② Self-Attention、Malti-head Attention 機構を取り入れて、単語の重層的な文中参照状況を検知・活用、③ Attention 機構が関連付けた情報をフィードフォワード NN で処理、④これらから成るブロックを多数重ね上げることで、序盤のブロックでは表記的な、中盤では文法的な、終盤では意味的な処理を多重実装（同論文では 6 回ループする）、⑤インプットの分散表現に文章中の単語の位置情報も追加し、前後関係の情報を強化、⑥残差接続[19] によって勾配消失問題の緩和や位置情報の保持（Attention 機構を通る際に情報がロストする点を迂回補強）を達成、といった特徴を有している（番号は図表 14 中の付番箇所に相当）。

図表14　Transformerモデルの構成概要



出所）Vaswani et al.（2017）
注）現論文の図表に筆者が補記（赤字赤線部分）

以下では、Attention 機構のコアとなる Scaled Dot-Product Attention について解説する[20]（Dot-Product は内積、Scaled は基準化）。図表 15 に計算過程を示している。

19：残差接続 / 残差結合（residual connection）とは、ひとつ前の層の結果を現在の層の出力結果に足し合わせる処理である。層 i への入力値を P_i、出力を O_i とすると、$P_i=O_i+P_{i-1}$ となる。出力結果に元の入力値を変形すると $P_i=O_i+O_{i-1}+O_{i-2}+O_{i-3}+\cdots+O_0$ となり、下層の出力を加算していったものが上層への入力値となる。残差という表現は、出力値が入力値の差となる点（$O_i=P_i-P_{i-1}$）から生じていると思われる。残差接続は、勾配消失問題の軽減のほか学習の安定化（一度に大きく変化させない）という効果を有する。

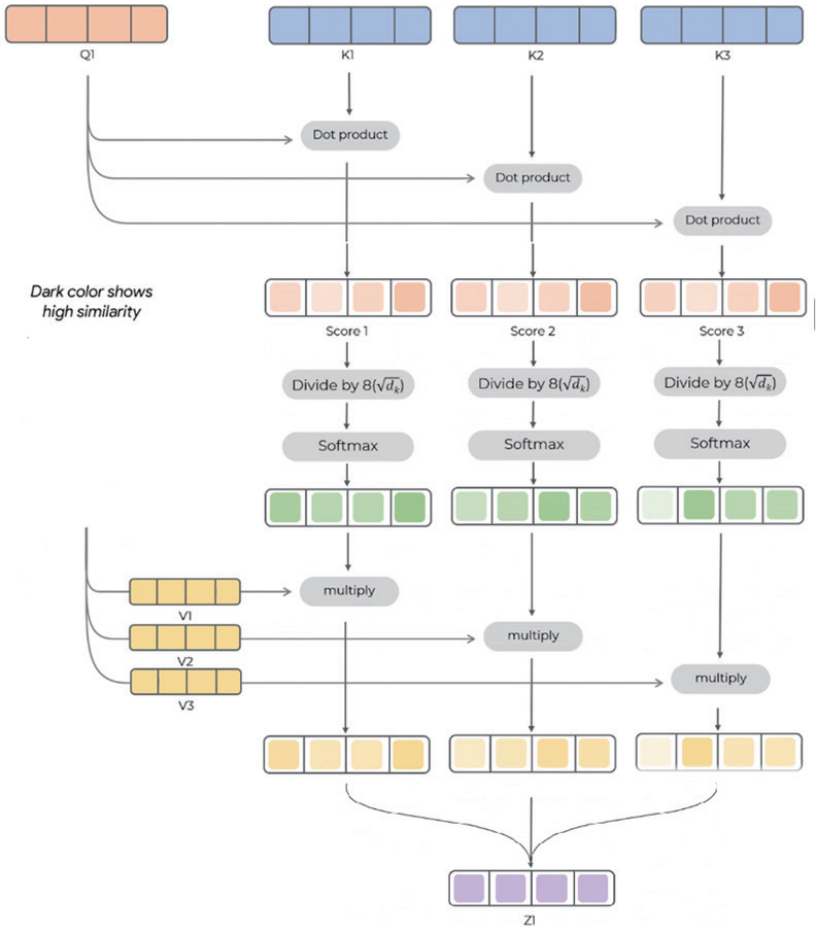
20：Attention 機構のアイディアは Transformer モデルが登場する前から Bahdanau, Cho and Bengio（2014）によって提唱されていたが、注目を集めたのは Transformer モデルに Scaled Dot-Product Attention の QKV 機構として組み込まれてからである。一般に Attention 機構という場合は、Transformer モデルに組み込まれた Scaled Dot-Product Attention と Multi-head Attention を指す。

まず、クエリ（Q）、キー（K）、バリュー（V）という概念を導入する。クエリ（問い合わせ）は、ある単語を対象にしたとき、文中の他の単語に対してどのような関連性があるかを調べる際の問い合わせ内容である。キーは、この問い合わせを受ける内容（関連性が評価される対象内容）である。クエリとキーの内積を計算することで関連性の強さを計測する（内積が類似性を示していることは 2.3 節の word2vec を参照）。これに softmax 関数を適用することで、内積をとった結果を確率表現化する。softmax 関数は、和が 1 となるよう入力値を変換するものであり、NN 全般においてアウトプットを確率として表現する際に用いられる（2.1 節で触れた「東京」が選ばれる確率や他の単語が選ばれる確率を計算する際にも同関数が利用されている）。この確率をバリューに対して乗じる。バリューは、インプットを分散表現（位置情報エンコーディングを加算）したベクトルの加重和であり、この中のどれに重きを置いて取り出すかを、Q と K の内積を通じて得た関連性情報に基づいて決定する。これが、Q、K、V 概念を使った Scaled Dot-Product Attention の仕組みである（図表 15 参照）。

ちなみに、Q、K、V のどの要素もインプットの分散表現をそのまま使うのではなく、その加重和を利用する。いずれもが同じ情報セットから加重和の違いによって作られるという発想は、LSTM の解説で示した全ての情報が h_{t-1} と x_t の加重和で更新されていくという NN の一般的な発想（全情報の中にある何らかのパターン性を学習によって発見して活用する）と同じである。一つ前のパラグラフで、バリューは、インプットの分散表現ベクトル「の加重和」と表記しているが、これは、問い合わせ（クエリ）との関連性を評価するキーと、キーの内容を示すバリュー、これらに出てくる Q、K、V の全てをインプットの線形結合で表現している仕組みが採用されているためである。インプット文章を構成する単語列という文書の内部構造に対して自己調査的に Attention を計測するため、Self-attention と呼称されている。

なお、デコーダー部分の下位の Attention 機構では、デコーダーが生成したアウトプットを順次利用するが、その先のアウトプット部分が先読み参照されないよう情報が遮断（mask）されている。また、上位の Attention 機構では、QKV として三本の矢印が異なる箇所から入力されている。エンコーダーのアウトプットからキーとバリューの元となる情報を受け取り、下層の Attention 機構のアウトプットからはクエリの元となる情報を受け取っている。これは、デコーダーにおいて作成済みのアウトプット（単語）が、次にくる単語の適切な選択を考える際のクエリ（問い合わせ）起点になるためである。それが指す対象であるキーや内容に相当する V はエンコーダーで抽出された情報を参照している。これが、上位の Attention 機構の QKV の特徴となっている。図表 12 の seq2seq モデルでは、デコーダーの一期前アウトプットを今期のインプットとし、エンコーダーから受け継いだ状態ベクトルをアップデイトしていっているが、これをより複雑に設計したエンコーダー・デコーダーモデルとなっている。

図表15　Scaled Dot-Product AttentionにおけるQKV計算



出所）Harsoor（2023）

注）QとKの内積に対して次元数d_k（ここでは64）のルートをとったもの、すなわち８で除算をしているのは、次元数が高いほど内積値が大きくなるため、その効果を基準化したものである。オリジナルのTransformer論文で導入されている。なお、Transformerモデルの動き方をExcelで追ってみることができる研究サイトがある。GPT-2を小規模にしてAttention機構の動きを数値例でモニターすることができる。

Ishan Anand, “Spreadsheets are all you need,” https://spreadsheets-are-all-you-need.ai/, (GitHub) https://github.com/ianand/spreadsheets-are-all-you-need/

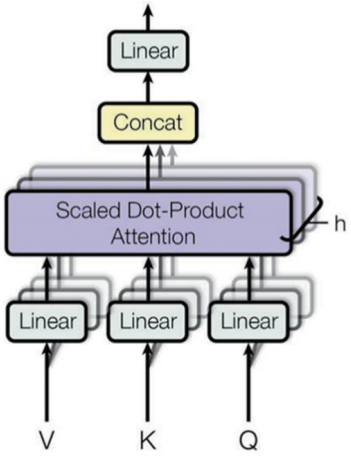
ところで、文中のある単語が指す、あるいは強い関連性を持つ単語は複数ある。関連性を捉える視点も複数存在する。これらに対応するためにはAttention機構を複数用意する必要がある。Transformerモデルを提唱したVaswani et al.（2017）では、インプットの各単語を512次元ベクトルに分散表現しているが、これを８つのグループに分けて、各64次元のベクトルに対しScaled Dot-Product Attentionを適用することで、複数の関係性を把握しようとしている[21]。これがマルチヘッドアテンション（Multi-head Attention）の仕組みとなっている。図表16にはh個のAttentionの結果が、単純に連結（concat）されて、更に重みづけ調整（図中の最上位のLinear）される仕組みが示されている。これが図表14中の残差接続と正規化作業の直前の箇所への出力となる。

21：８グループへの切り方は512次元の前方から順に切っていくだけであるが、そのように分断された分散表現に対して個々にAttention機構を適用するとAttentionのパフォーマンスが低下すると考えられる。しかし、実際にはTransformerモデルの性能は大きく改善しており、関係性の発見を多様な視点で与えることのメリットが上回っているものと思われる。

図表 17 にはマルチヘッドアテンションが実際にどのように単語間の参照関係を計測しているか調査した事例を紹介している。左側では、名詞句に形容詞句がどのように修飾関係をもってかかってくるかを探知する Attention が作成されている。右側は受け身関係にある be 動詞＋過去分詞の関係を探知する Attention が作成されている。こうした様々な Attention はモデルの学習（NN における学習）によって獲得される。

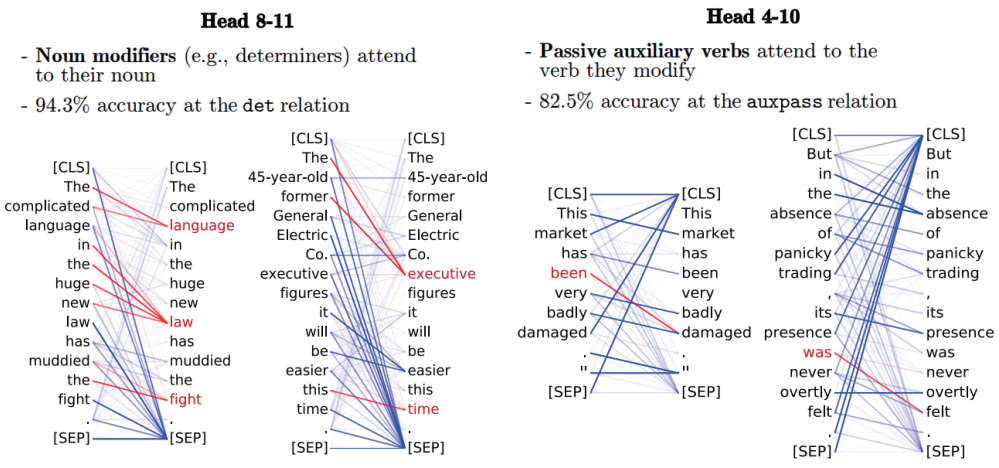
Transformer モデルは、RNN のような入れ子構造で単語間の関係を調べるのではなく、Attention 機構を活用するため並列処理が可能となり、計算負荷の軽減やモデルの大規模化に繋がった。マルチヘッドアテンションも、分散表現によるベクトル次元サイズを分割することで並列処理を更に強化している。

図表16　マルチヘッドアテンション



出所）Vaswani et al.（2017）

図表17　マルチヘッドアテンションの事例



出所）Clark et al.（2019）

Transformer モデルは Attention 機構に注目が集まりがちだが、単語間の関連性や指示関係性が組み込まれた状態ベクトルは、フィードフォワード NN のパラメータ群で整理される。言い換えると、学習された情報はこのパラメータ群に蓄積され、知識となる。Attention 機構を含むエンコーダー側のブロック（図表 14 のグレー部分）は 6 回積み重なったうえで、デコーダー側に情報を引き渡すが、上位階層のフィードフォワード NN ほど高次な意味概念を蓄積する役割を果たしている。また、同 NN は、インプットの次元が 512 であるのに対し、中間層が 2048 と 4 倍増しており、特徴量抽出の機能が強化されている（ただし中間層は 1 つのみである）。

3.5 LLMの群雄割拠

Transformer モデルの登場は、ニューラル言語モデルの能力を劇的に高めた。最初に注目を集めたのが Google の研究チーム、Devlin et al.（2018）によって考案された BERT（Bidirectional Encoder Representations from Transformers）である。BERT や同年登場した GPT は次節で触れる事前学習（Pre-training）とファインチューニングのパラダイムを確立し、その後の LLM の発展方向に大きな影響を与えた。また、BERT は Transformer モデルのうちデコーダー部を利用せず、エンコーダー部のみを用いた点も特徴的である。その代わり、分散表現の次元数が 768、ブロックのループが 12 回もしくは 24 回（BERT Large モデル）、マルチヘッド数は 12 もしくは 16 とオリジナルの Transformer モデルより規模が大きくなっている。BERT の成功により、デコーダー系のモデルとして RoBERTa、DistilBERT、ALBERT、XLNet など様々なバリエーションや改良版が登場した。

同 2018 年には、その後の生成 AI/LLM ブームを引き起こした GPT（Generative Pre-trained Transformer）の初代モデルが OpenAI 社によって開発された（論文公表時でみると GPT が先行している）。BERT と同様、その名前には Transformer が含まれている。2010 年代を通じてニューラル言語モデルの開発拠点が大学から企業へシフトしつつあったが、この時期になると、開発の主導がプラットフォーマー大企業や IT 企業の研究チーム中心となる[22]。そうした LLM モデル群が登場した 2018 年から 19 年にかけては、モデルの大規模化が起こり始めた（更に加速したのはその後であり、次節で解説する）。LLM（大規模言語モデル）という言葉は、Transformer モデル登場後のモデルの大規模化とこれに伴う性能向上によって誕生し、実務への応用が進んだことによって一般に知られていくことになる。

図表 18 は、そうした LLM の進化を樹形系統図で示している。2023 年初までの状況であり、例えば Google の PaLM2 を引き継いだ Gemini は図中には登場していないが、Transformer 以降の LLM 時代の始まりを概観するのに有益である。同図は、2018 年の段階で Transformer モデルが 3 つに分岐したことを示している。エンコーダーモデル（樹形図の左側）、デコーダーモデル（右側）、エンコーダー・デコーダーモデル（中央）である。BERT がエンコーダーの代表例であり、2019 ～ 20 年にかけて派生拡張モデルを生んだが、その後は続いていない。LLM 開発においては文章理解に加えて文

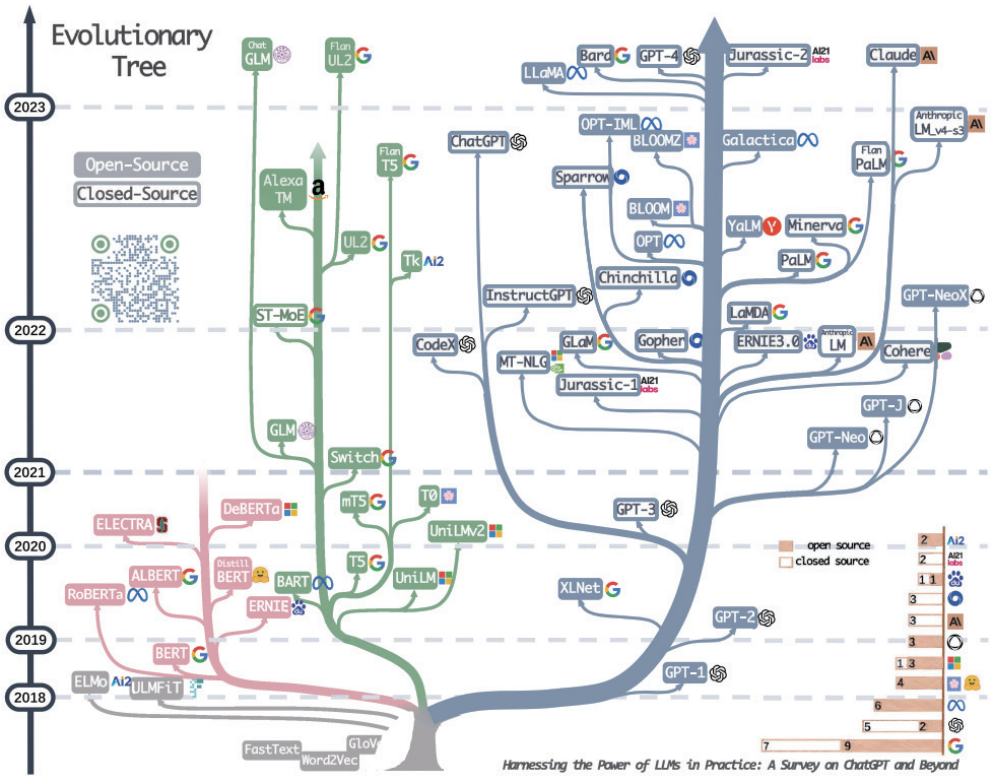
22：本稿の参考文献を Google scholar 等で検索すると論文を参照できる。表紙の筆者所属情報を時代順にみていくと、こうしたシフトが観察される。また、共著者数の増加も企業主導型になったことの反映であると思われる。執筆者が 10 名を超える論文も少なくない。

章を生成する機能も重視され、また、画像や音声の生成などマルチモーダル化が進んだため、復号機能を持つデコーダー系に数多くの LLM が登場した。OpenAI の GPT シリーズもデコーダー系である。そもそも、デコーダーモデルは抽象化された状態ベクトルを生成する（自然言語など元のものに復号しない）。このため、文書分類や感情分析などの選択問題、文章中の特定の単語やフレーズが何に相当するかを識別する固有表現抽出、文書間や文間の関係性判断（例えば整合的 / 矛盾 / 無関係）など、用途が限定されているという事情もあった[23]。

BERT で文章生成を行う場合は、デコーダー型の別のモデルを組み合わせるが、文章生成を目的とするならば、オリジナルの Transformer モデル通りにエンコーダー・デコーダーの一体型モデルで開発したほうが効率的である。この発想で登場したのが Facebook/Meta の BART（Bidirectional Auto-Regressive Transformer）であり、ほぼ同時期に Google も T5（Text-to-Text Transfer Transformer）を開発している（図中央の枝を参照）。

23：そうはいっても、技術的スタックは今も有用である。例えば、RAG では LLM の性能以上に情報データベースからの検索性能にサービスの質が依存するため、ベクトル化データベースに対する類似度検索でなく BM25 のような伝統的検索インデックス（TF-IDF の組み合わせ活用、詳細は打田他（2022）を参照）が重要となる。BM25 に BERT のアウトプットの一つである CLS（言語から符号化された状態ベクトルの一部で、意味情報などを集約している）を加味することで、検索能力を高性能化した BM42 が考案されるという動きが生じている（Vasnetsov 2024）。BM25/TF-IDF は 40 年前から利用され続けている技術であるが、そうした既存の技術との組み合わせなど LLM には様々な可能性が秘められている。

図表18　LLM進化の系統図



出所）　Clark et al.（2019）

Google は、BERT から T5 に移行し、その後は、PaLM/Gemini シリーズでデコーダーモデルに開発をシフトさせている。Facebook/Meta も、BART や BERT の発展形 RoBERTa から、2022 年以降は LLaMA シリーズや OPT のようにデコーダーモデルにシフトしている。Meta は、OpenAI や Google と異なりオープンソース化戦略をとっている点に特徴がある。

Google も Flan-T5 や Gemma など一部をオープンソース化している。Meta 社が OSS 提供している LLaMA シリーズをベースに開発を進めた LLM も多く登場しており、例えば、日本の直近の事例では ELYZA 社が Llama-3-ELYZA-JP-70B/8B を開発・公開している（70B はパラメータ数が 700 億であることを示している）[24]。

LLM を整理する場合、エンコーダー型かデコーダー型かというモデル構造上の視点のほか、オープンソースかクローズドか、大規模モデルか小規模モデルか[25]、学習方法のバリエーション、言語特化型か画像や音声にも対応したマルチモーダルか、学習データの性質（文章、プログラム言語、音声等）と量・質、API 対応の有無（商用は API 対応）、有償 / 無償、ファインチューニングの可否（次節参照）など、様々な視点がある。

24：最近のオープンソース LLM では、フランスの AI スタートアップ Mistral の Mixtral、UAE のテック企業 TII の Falcon、MosaicML の MPT、Hugging Face · BigScience プロジェクトの BLOOM、Databricks の DBRX、xAI（イーロン · マスクの X 関連企業）の Grok などが知られている。

25：LLM の大規模化が著しいため、大規模か小規模かは相対的な差に過ぎないが、ローカル LLM として PC やスマートフォンなどのデバイスでエッジコンピューティングが可能になるかという視点では、ハードウエア対比の絶対水準が意味を持つ。小規模 LLM のエッジコンピューティングは今後の発展方向として注目されている。最近の LLM は中小規模のものをシリーズで展開するケースが少なくない。大規模モデルで学習した内容を小規模モデルに転移させる技術として知識蒸留（Knowledge Distillation）と呼ばれる手法が発展しており、小規模モデルのラインナップ拡充に寄与している。

図表19　主要なLLM群

Type	Model Name	#Parameters	Release	Base Models	Open Source	#Tokens	Training dataset
Encoder-Only	BERT	110M, 340M	2018	-	✓	137B	BooksCorpus, English Wikipedia
	RoBERTa	355M	2019	-	✓	2.2T	BooksCorpus, English Wikipedia, CC-NEWS, STORIES (a subset of Common Crawl), Reddit
	ALBERT	12M, 18M, 60M, 235M	2019	-	✓	137B	BooksCorpus, English Wikipedia
	DeBERTa	-	2020	-	✓	-	BooksCorpus, English Wikipedia, STORIES, Reddit content
	XLNet	110M, 340M	2019	-	✓	32.89B	BooksCorpus, English Wikipedia, Giga5, Common Crawl, ClueWeb 2012-B
Decoder-only	GPT-1	120M	2018	-	✓	1.3B	BooksCorpus
	GPT-2	1.5B	2019	-	✓	10B	Reddit outbound
Encoder-Decoder	T5 (Base)	223M	2019	-	✓	156B	Common Crawl
	MT5 (Base)	300M	2020	-	✓	-	New Common Crawl-based dataset in 101 languages (m Common Crawl)
	BART (Base)	139M	2019	-	✓	-	Corrupting text
GPT Family	GPT-3	125M, 350M, 760M, 1.3B, 2.7B, 6.7B, 13B, 175B	2020		×	300B	Common Crawl (filtered), WebText2, Books1, Books2, Wikipedia
	CODEX	12B	2021	GPT	✓	-	Public GitHub software repositories
	WebGPT	760M, 13B, 175B	2021	GPT-3	×	-	ELI5
	GPT-4	1.76T	2023	-	×	13T	-
LLaMA Family	LLaMA1	7B, 13B, 33B, 65B	2023	-	✓	1T, 1.4T	Online sources
	LLaMA2	7B, 13B, 34B, 70B	2023	-	✓	2T	Online sources
	Alpaca	7B	2023	LLaMA1	✓	-	GPT-3.5
	Vicuna-13B	13B	2023	LLaMA1	✓	-	GPT-3.5
	Koala	13B	2023	LLaMA	✓	-	Dialogue data
	Mistral-7B	7.3B	2023		✓	-	-
	Code Llama	34	2023	LLaMA2	✓	500B	Publicly available code
	LongLLaMA	3B, 7B	2023	OpenLLaMA	✓	1T	-
	LLaMA-Pro-8B	8.3B	2024	LLaMA2-7B	✓	80B	Code and math corpora
	TinyLlama-1.1B	1.1B	2024	LLaMA1.1B	✓	3T	SlimPajama, Starcoderdata
PaLM Family	PaLM	8B, 62B, 540B	2022	-	×	780B	Web documents, books, Wikipedia, conversations, GitHub code
	U-PaLM	8B, 62B, 540B	2022	-	×	1.3B	Web documents, books, Wikipedia, conversations, GitHub code
	PaLM-2	340B	2023	-	✓	3.6T	Web documents, books, code, mathematics, conversational data
	Med-PaLM	540B	2022	PaLM	×	780B	HealthSearchQA, MedicationQA, LiveQA
	Med-PaLM 2	-	2023	PaLM 2	×	-	MedQA, MedMCQA, HealthSearchQA, LiveQA, MedicationQA
Other Popular LLMs	FLAN	137B	2021	LaMDA-PT	✓	-	Web documents, code, dialog data, Wikipedia
	Gopher	280B	2021	-	×	300B	MassiveText
	ERNIE 4.0	10B	2023	-	×	4TB	Chinese text
	Retro	7.5B	2021	-	×	600B	MassiveText
	LaMDA	137B	2022	-	×	168B	public dialog data and web documents
	ChinChilla	70B	2022	-	×	1.4T	MassiveText
	Galactia-120B	120B	2022	-		450B	
	CodeGen	16.1B	2022	-	✓	-	THE PILE, BIGQUERY, BIGPYTHON
	BLOOM	176B	2022	-	✓	366B	ROOTS
	Zephyr	7.24B	2023	Mistral-7B	✓	800B	Synthetic data
	Grok-0	33B	2023	-	×	-	Online source
	ORCA-2	13B	2023	LLaMA2	-	2001B	-
	StartCoder	15.5B	2023	-	✓	35B	GitHub
	MPT	7B	2023	-	✓	1T	RedPajama, m Common Crawl, S2ORC, Common Crawl
	Mixtral-8x7B	46.7B	2023	-	✓	-	Instruction dataset
	Falcon 180B	180B	2023	-	✓	3.5T	RefinedWeb
	Gemini	1.8B, 3.25B	2023		✓	-	Web documents, books, and code, image data, audio data, video data
	DeepSeek-Coder	1.3B, 6.7B, 33B	2024	-	✓	2T	GitHub's Markdown and StackExchange
	DocLLM	1B,7B	2024	-	×	2T	IIT-CDIP Test Collection 1.0, DocBank

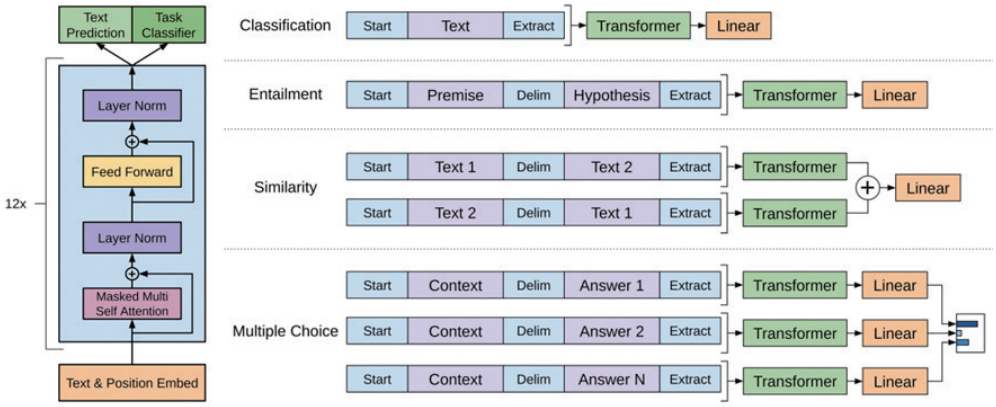
出所）Minaee et al.（2024）

図表 20 は GPT-1 のデコーダーモデルを示したものである[26]。Attention 機構の内部は同じであり、12 階層となっている。デコーダーの Attention 機構では未来情報にマスクがかかっているため、既出である単語について過去方向にしか Attention を向けることができない分、学習能力は低下するが、BERT と異なり文章などへの復号が可能という利点がある。単語の分散表現に BPE（Byte Pair Encoding）と呼ばれる手法を用いており、単語を更にサブワードに分割してトークン化し、これを分散表現の対象とするものである（例えば、sub/word、富士 / 山、もしくは富 / 士 / 山）[27]。

26：図表の右側では、インプットとして 2 文を接続させ、セパレーターで挟み込んで使う方法が示されている。これは、同時期に開発公表された BERT も同様な作りとなっている。文書分類なら 1 文のみ、命題と仮説からの含意は 2 文の組み合わせ、類似性評価なら 2 系統構成というように使い分けが示されている。

27：日本語の文章を分散表現にかけると、富 / 士 / 山のように、ほぼ一文字が 1 トークンになるまで分解される。英語のように単語間でスペースが存在する言語に比べ、切れ目がない言語は一般に非常に細かくトークン化される傾向がある。一文字が複数のトークンで表現されることもある。

図表20　GPT-1のデコーダーモデル



出所）Radford et al.（2018）

LLM 開発においては性能評価が必須であり、評価手法についても研究開発が進んでいる。図表 21 は、AI/ML 開発者にモデル開発プラットホームを提供する Weights & Biases の日本法人が提供している日本語を対象としたリーダーボードである。llm-jp-eval（言語理解）と Japanese MT-bench（言語生成）で評価されていた「汎用的言語性能」に加え、「アラインメント」という新たな評価軸が導入され、総合評価上位順にリストアップされている。Weights and Biases Japan（2024）はこれらの評価軸について解説しており、汎用的言語能力が翻訳や推論、検索、知識・質問応答、意味解析など様々な視点からの能力評価の合成値であることを解説している。アラインメントは、制御性、倫理・道徳、毒性、バイアス、真実性、堅牢性などといった公正性（フェアネス）などに係る視点からの評価がなされている。

図表 21 は、同社のサイトに掲示されているリーダーボードであり、ユーザが表示形式を操作可能な UI（ユーザインターフェイス）が採用されている。横軸に様々な評価基準が並んでおり、汎用的言語性能でリストアップすると、上位には Anthropic 社の Claude3.5 sonnet や OpenAI 社の GPT-4o、Gemini 1.5 Pro などの 2024 年 7 月初時点の最新の商用 LLM がランクインしていることが見て取れる。図表 22 は同社のブログ（note）に掲載されたものであり、上記 2 軸で評価された LLM が x-y プロットされている。グローバル企業の商用 LLM（ほとんどは API 提供されており外部から使用）が両軸と

もに上位となっていることがわかる。ランキングは日進月歩で変化しており、同社 website で時系列プロットしてみると性能向上が日々続いていることがわかる。

図表21　Weights & Biases JapanのNejumi LLMリーダーボード3

GLP : General Language Processing (汎用的言語性能)

ALT : Alignment (アライメント)

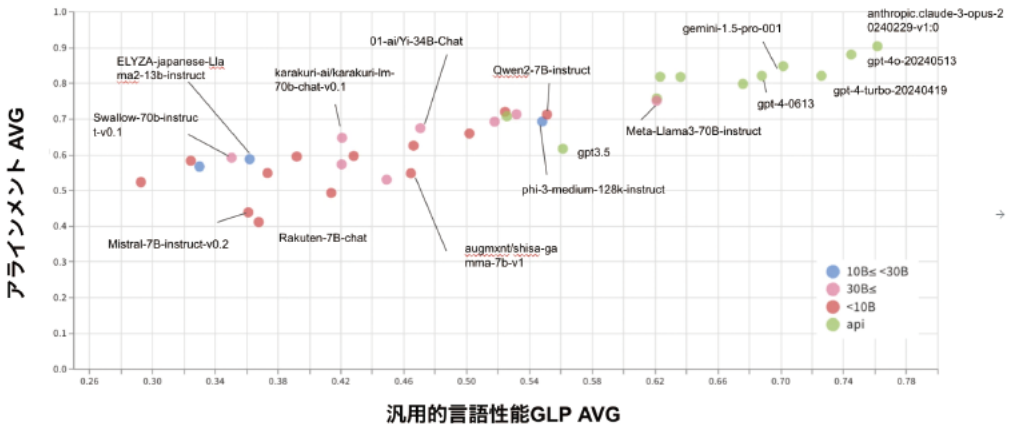
Total AVG = (Avg. GLP + Avg. ALT)/2

runs.summary["leaderboard_table"]

	model_name	model_size_cz	汎用的言語性能(GLP)_AV	アラインメン	TOTAL_AVG	GLP_表現	GLP_翻訳	GLP_情報検索	GLI
12	anthropic.claude-3-5-sonnet-20240620-	api	0.7618	0.9027	0.8322	0.8733	0.8684	0.8061	
6	gpt-4o-2024-05-13	api	0.7451	0.8796	0.8123	0.895	0.8693	0.8227	
35	gpt-4-turbo-2024-04-09	api	0.7019	0.8467	0.7743	0.8917	0.8607	0.813	
13	anthropic.claude-3-opus-20240229-v1:0	api	0.7262	0.82	0.7731	0.885	0.872	0.7899	
15	gemini-1.5-pro-001	api	0.6882	0.8199	0.754	0.88	0.8534	0.8019	
36	gpt-4-0613	api	0.6762	0.7973	0.7367	0.8133	0.8587	0.8926	
10	gemini-1.5-flash-001	api	0.6365	0.817	0.7268	0.8667	0.8452	0.7472	
9	anthropic.claude-3-sonnet-20240229-	api	0.6235	0.8176	0.7205	0.8567	0.8313	0.748	
14	anthropic.claude-3-haiku-20240307-v1:0	api	0.6214	0.7564	0.6889	0.8183	0.8045	0.7991	
31	meta-llama/Meta-Llama-3-70B-Instruct	30B≤	0.6213	0.7499	0.6856	0.7533	0.8516	0.8753	
1	cyberagent/calm3-22b-chat	10B≤ <30B	0.625	0.7164	0.6707	0.8517	0.8431	0.8825	

出所）Weights & Biases Japan Website 2024年7月5日時点

図表22　日本語LLMリーダーボード



出所）Weights & Biases Japan（2024） https://note.com/wandb_jp/n/nd4e54c2020ce

4．LLM発展過程での発見

4.1 3つのスケール測と規模拡大競争

学習や推論において並列処理が可能となった Transformer 型のモデルでは、モデルの大規模化が進展した。Attention 機構により文章上、遠く離れた箇所どうしの関係性が補足できるようになったため、長文をまとめて処理する利点を活用したいという誘因も大規模化を促した。インプット数の増大は、DNN の上部の層が深いほどパラメータ数を乗数的に増加させる。ブロックの繰り返しによりパフォーマンスを引き上げられる点もディープ化を促した。また、そもそも Attention 機構の QKV の仕組みは大量にパラメータを使用するものであった。一方で、並列処理による計算負荷の軽減が、大規模化に伴う計算制約を緩和する効果もあった。

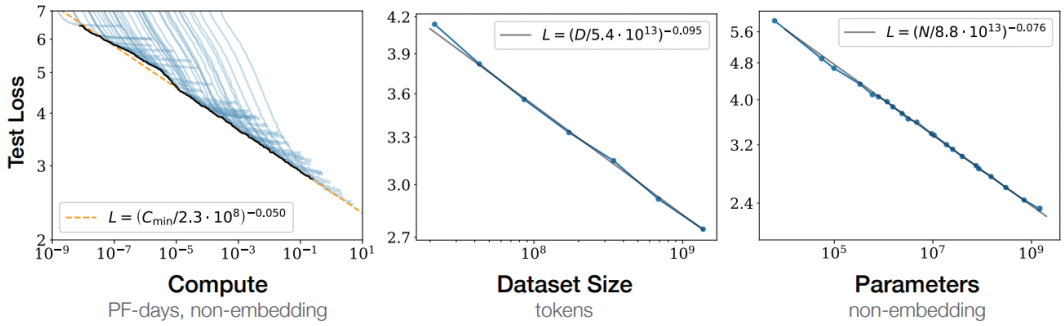
このようなモデル発展上の要請のほか、GPU などハードウエアの進歩、クラウドコンピューティングの普及といった計算資源の拡充や、分散コンピューティングの技術進歩による分散学習（並列処理の利点）、学習用データセットの整備（Common Crawl などを活用したインターネット上の情報収集など）もモデルの大規模化を促した。

Kaplan et al.（2020）は、OpenAI の研究者 10 名による規模とパフォーマンスの関係に関する調査であり、計算資源、学習データ、モデルサイズ（パラメータ数）の 3 要素を大規模化させた場合、どの程度のパフォーマンス改善が図れるかを各要素について検証している。同論文は、図表 23 に示された 3 つのスケール則を発見している。右図は、モデルを大規模にしてパラメータ数を増やすほど、べき乗則に従ってモデルのパフォーマンスが高まることを示している（縦軸の損失関数が線形に減少[28]）。中央図は学習に用いるデータセットの規模についてもべき乗則のスケール則が成立していることを示している。左図の計算資源を対象とした検証図には複数の右下がり線が描かれている。図中の右側の線は、パラメータ数が多いモデルに計算資源をより多く投入した場合のパフォーマンス改善を示している。線形に改善していくが一定値までくると限界に直面し、右横方向に推移している（正確には緩やかな右下がりに変化）。左側の中小規模モデルは、一定のパフォーマンスを達成するには大規模モデルより計算資源を必要としない（y 軸の一定値で比較）が、改善の限界に直面するのも早く、それ以上の改善を望むならモデルを大きくするしかないことを示している。この限界線を様々な規模のモデルについて結ぶと右下がりの直線（黒太線）が現れており、計算資源についてもべき乗則が成立していることがわかる。

この発見は、LLM の開発方針に大きな影響を与えた。大規模化すれば確実に性能が改善することが、べき乗則が成立し続ける間は保証されている。図表 24・25 は GPT シリーズのパラメータ数の拡大を示している。GPT-2 から GPT-3 に向けてパラメータ数が 100 倍以上に引き上げられ、学習データもより巨大なものを用いるようになり、学習に巨大な計算資源を振り向けられるようになった。この時期に、LLM の開発における規模拡大競争が明確なものとなった。

28：べき乗則は、べき分布（power law distribution）に従って「ある値が他の値に対してどのようにスケーリングするか」を示すものである。べき分布は $P(x)=cx^{-\alpha}$（α はべき指数）と表現され、これを対数表示すると、$\log(P(x))=\log(C)-\alpha\log(x)$ となる。同式は、x が 1%増加すると $P(x)$ が α %減少することを意味している。図表 23 は、表示された領域全域で x の規模にかかわらずべき乗則が維持されていることを示している。なお、べき乗則は、家計の所得や資産分布、市町村の人口規模の分布、書籍の販売部数の分布、資産価格変動率の分布といった社会現象だけでなく、自然現象においてもしばしば観察され、フラクタル理論とも関連性が高い概念である。例えば、与信ポートフォリオの集中リスク計測においても、企業規模 / 与信規模の分布は重要な要素となる。金融市場の価格変動にもべき乗則やフラクタル特性が観察されている。

図表23　3つのスケール測



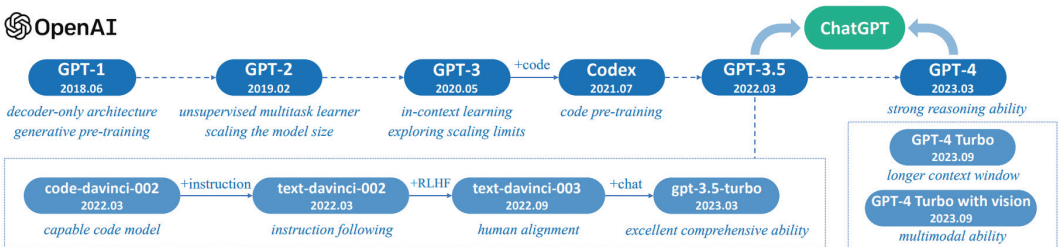
出所）Kaplan et.al.（2020）

図表24　GPTシリーズの規模拡大

モデル	パラメータ数	公開年月	主な特徴、開発論文
GPT-1	1.17 億	2018/6	Transformer のデコード型モデルとして言語生成を行う事前学習（pre-training）型の初の大規模言語モデル。Radford et al.（2018）
GPT-2	15 億	2019/2	Attention 機構を 48 層としモデル規模を拡大させたほか、大規模データセットで学習することにより、一貫性がある文章が生成されやすくなるなど能力が向上。Radford et al.（2019）
GPT-3	1,750 億	2020/5	スケール則の発見を活かすべくパラメータ数を一気に約 100 倍増させ、言語生成能力が大きく向上。Common Crawl という大規模データセット（後述）を学習に利用している。In-context learning が可能となった。Brown et al.（2020）
GPT-3.5	非公開	2022/3	改良されたトレーニング方法とデータセットにより、性能と応用範囲が更に向上し、ゼロショットでの回答精度やアラインメントの質が高まったことから対話サービス ChatGPT のバックエンドで利用されるようになった（図表 25 参照）。
GPT-4	非公開	2023/3	マルチモーダルに対応。精度と安全性を向上させるため、強化学習と人間によるフィードバックを利用している。Open AI et al.（2023）

出所）各開発論文より

図表25　GPTシリーズの発展とサブライン（下段）

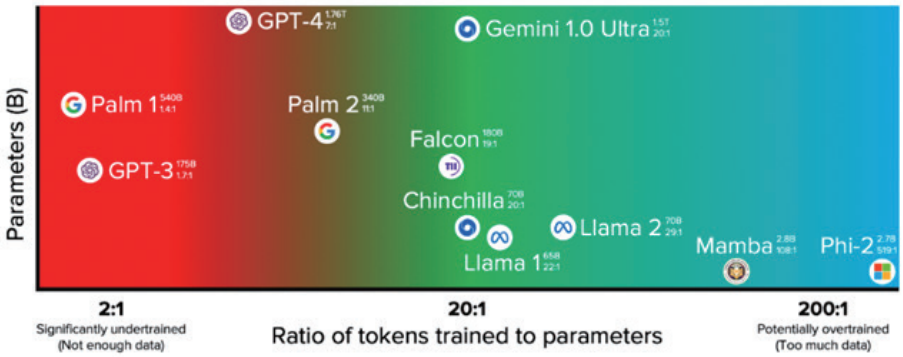


出所）Zhao et al.（2023）

前述のスケール則を前提とすると、3 つの要素のうち何を拡張すると効率よく性能向上が図れるかという問題に直面する。現状を前提に改善するときは、スケール則に従ってパラメータと学習データセットの増加率を同じにすればよい。しかし、現在の状態がベストなバランスである保証はない。DeepMind の研究者たちは、Hoffmann et al.（2022）において、計算資源を一定とすれば、最良のモデルは必ずしも大きいモデルではなく、比較的小規模なモデルを比較的大量のデータで学習したモデルとなることを示し、当時加速しつつあったモデルの巨大化開発方針が適切でない可能性を指摘した。同論文は、トークン（単語）数とパラメータ数の比率は 20：1 が望ましいとしている。この目安の算出には同社が開発した LLM の Chinchilla を検証に用いたため、チンチラのスケーリング則と呼ばれるようになった。その後、Sardana and Frankle（2023）は、チンチラ則は学習時の計算資源制約を前提としたものであり、実際に使用する場合の推論コスト（inference cost）は考慮していないという指摘を行い、更にモデル規模を小さくしたほうが望ましい（190：1）という検証結果を得ている。実用化の際には、ランニングコストが重要になるため、推論コストを検討対象に入れることは重要である。また、生成速度というレスポンス時間の面でも最適規模の選定は実用化の際の重要な要素となる。

ただし、その後も、先端的モデルの大規模化は継続しており、GPT-4 は非公表ながら数千億から 1 兆を上回っているという推測がなされている（図表 26 では 1.76 兆という数字が示されている）。他の主要モデルも大規模化を図っており、図表 27 のリーダーボード（日本語評価）の上位に出てくるようなモデルは数百億から数千億のオーダーとなっている。図表 26 に示した主要 LLM のトークン数とパラメータ数のバランスもチンチラ則からは様々に乖離している。例えば、大幅な大規模化を目指した GPT-3 は 20：1 ではなく 2：1 でモデル規模を優先させている。

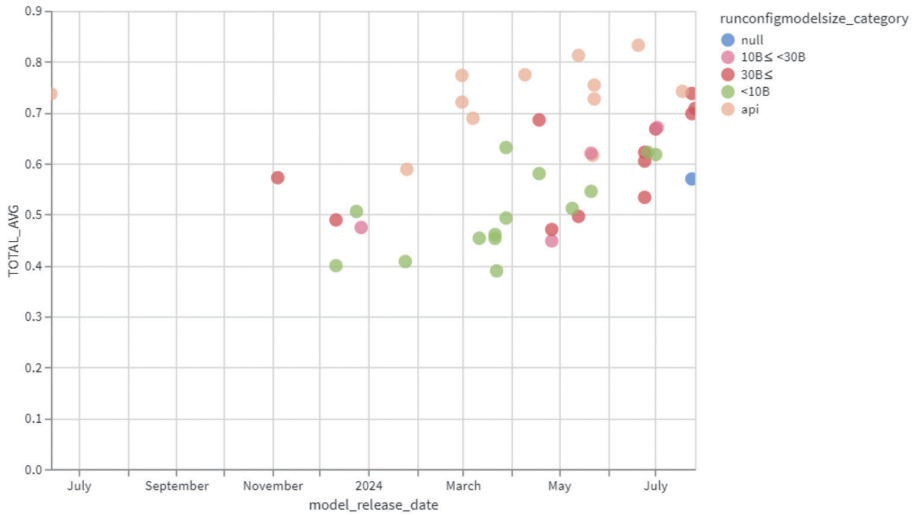
図表26　チンチラ則との乖離



出所）Thompson（2024）

一方で、コスト対比でのパフォーマンスを優先させるため、ハイエンドモデルを知識蒸留などにより小型化したモデルも多く登場している。オープンソースの LLM を専門分野知識特化型や特定言語特化型に拡張開発するため、別学習セットを用いて追加学習させる研究も盛んに行われている。そこでも開発・学習コストの面で、数十～数百億程度の中小型モデルが採用されている。日本語を強化したモデルとしてリーダーボードに登場している開発例をみても、サイズを抑制する代わりに学習データセットとチューニングやアラインメントで性能を上げる方針のものが多い。前述したリーダーボードのサイトで確認すると ELYZA/Llama-3-ELYZA-JP-8B や、Cyberagent/calm3-22b-chat、KARAKURI-AI/karakuri-lm-8x7b-chat-v0.1、TokyoTech-LLM/Llama-3-Swallow-8B-Instruct-v0.1 などである。これらの多くは、Llama-3 や Calm3 などオープンソースの最新 LLM をベースに追加学習が行われている。一方で、NEC や Preferred Networks、ABEJA などのようにスクラッチで LLM を開発している先もあり、これらについてもグローバルトップリーダーの最新 LLM と比較すると比較的規模が小さいモデルを用いて日本語対応性能を引き上げようとするものが多い。

図表27　Weights & Biases JapanのNejumi LLMリーダーボード3

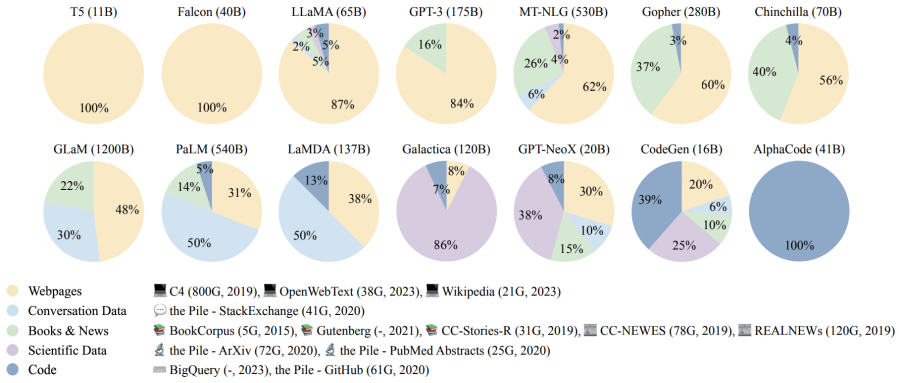


出所）Weights & Biases Japan, Nejumi LLMリーダーボード, 2024年7月31日取得

学習セットの大規模化は、量と対象の面において拡大している。まず、インターネット上にある情報をクローリングして集めてくる手法の高度化があげられる。Common Crawl は、ペタバイト規模のデータ量を含むオープンソースのウェブクロールデータベースであり、LLM の学習に広く利用されている。サブセットも整備されており、例えば、C4、CC-Stories、CC-News、RealNews などがある。C4（Colossal Clean Crawled Corpus）のなかにおいても、en（806G）、en.noclean（6T）、realnewslike（36G）、webtextlike（17G）、および multilingual（38T）の５つのバリエーションがある（カッコ内はバイト数表示＜ギガバイト、テラバイト＞、トークン化を行った後はトークン数で表記されることが多いが、コーパス段階ではバイト評

価が一般的）。このほか、各国語の Wikipedia も生コーパスとして利用されている。Reddit は、Facebook とほぼ同時期に誕生した米国発祥の SNS プラットホームで、匿名で書き込まれ米国版 2 ちゃんねるとも呼ばれているが、文章の質が全般に保たれているため、高品質のデータセットを作るための生コーパスの元になっている。著作権が切れた書籍も学習用のデータベースとして利用されており、BookCorpus や Project Gutenberg が知られている。コンピュータプログラムも学習対象であり、GitHub のようなコードレポジトリや、StackOverflow のようなコードに関する Q&A プラットホームも学習用データとして整備・活用されている。図表 28 は 14 の LLM について、学習データセットの構成比とサイズ（LLM に付されている数値はトークン単位、凡例のデータセットはビット単位）を示している。

図表28　学習データセットの巨大化と多様化

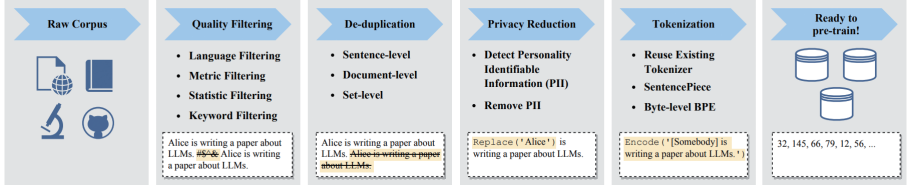


出所）Zhao et al.（2023）

規模以外にも学習データセットの質が LLM の性能に影響するため、図表 29 で示したような前処理が行われている[29]。生コーパスからの品質フィルタリング、重複の除去、プライバシー情報の削除（これはアラインメント目的）、html のタグ除去（一部には文脈上の意味があるものも存在）、分散表現を行う前のトークン化（文を単語や単語以下のサブワードに分割する）などが行われたうえで、分散表現技術によりベクトル化される。こうした過程で生コーパスの規模（ビット計測）は大きく縮約される。例えば、Penedo et al.（2023）を参照。また、学習データの質を高めるほうが規模を大きくするより遥かに効果的であるという実証研究もなされている（Gunasekar et al. 2023）。

29：日本語大規模ウェブコーパスについて同作業を行ったものとして岡崎他 (2024) がある。

図表29　学習データの生成



出所）Zhao et al.（2023）

4.2 様々な転移学習の発見：事前学習とファインチューニング

LLM の発展を理解するためには、転移学習が自然言語処理の発展に果たした役割を知ることが重要となる。転移学習とは AI や機械学習の発展において生じた概念であり、ある領域（ソースとなるドメイン）のタスクにおいて獲得された知識を別の領域（目標ドメイン）のタスクに転用することを指す。松井・熊谷（2024）が転移学習について包括的な解説を行っている。生成 AI/LLM における転移学習は、発展段階の様々なポイントで異なった形態で登場してきた。以下で順に解説する。

4.2.1 分散表現

最初の代表事例は、word2vec のような単語（トークン）の分散表現である。あるコーパスから学習によって得られた分散表現は、一定の汎用性をもって別の領域の自然言語処理にも適用できる。これは転移学習の一形態に相当する。前出の ELMo は文脈に依存して単語を分散表現する手法であり、こちらも文脈に関する獲得情報が転移学習可能であることを示している。LLM 開発において学習データセットは英語がほとんどであり他国語は少ししか含まれていない場合であっても、学習済みの LLM が他国語への対応能力を一定程度有することが知られている。言語の成り立ちに普遍性があるならば、主に英語から抽出された言語の特徴が他国語の自然言語処理にも一部転移可能であることを示唆している。

4.2.2 事前学習とファインチューニング

GPT-1 と BERT は、事前学習（Pre-training）とファインチューニング（Fine-tuning）というコンセプトを LLM に導入した[30]。自然言語理解や生成において転移学習が可能であれば、ある LLM が大きな学習データセットにおいて獲得した自然言語処理能力を別の LLM に転移することができる。この前半部が事前学習であり、その転移された能力を更に改善する行為をファインチューニングと呼ぶ。一般に、ファインチューニングにおいてはモデルのパラメータ群が更新されるため、追加的な学習データセットが比較的小規模であったとしても大きな計算資源を要することが多い。チューニングも容易ではなく、高精度・高性能に作りこまれた汎用 LLM の性能を逆に落としかねないリスクもある。機械学習、とりわけ NN では、新しい情報を学習する過程において以前に学習した情報が急激に失われる現象が生じることが知られている。これは破滅的忘却（Catastrophic Forgetting）と呼ばれ、LLM の事前学習プロセスやファインチューニングにおいても発生している[31]。こうした現象を回避し、かつ膨大な計算資源の消費を回避するために、より効率的な追加学習方法として PEFT（Parameter Efficient Fine-Tuning）と呼ばれる技術が開発されており後述する。

4.2.3 教師ありファインチューニング

より簡便な方法として、教師ありファインチューニング（Supervised Fine-Tuning）がある。事前学習されたモデルを特定のタスクやドメインに

30：Radford et al.(2018)、Devlin, Chang, Lee and Toutanova(2018) を参照。

31：2022 年にリリースされた GPT-3.5 ではユーザによる追加学習が可能となった。筆者は専門領域分野の日本語 Q&A100 問を用いて小規模な Supervised Fine-Tuning を GPT-3.5 で行ったことがある。専門知識の反映はある程度の精度を持って行えたが、回答文章の日本語品質が明らかに低下しており、「破滅的」忘却ほどではないが汎用 LLM の機能が一部劣化することを経験した。追加学習にかかるコスト（計算負荷に応じてチャージされる利用料金）も推論に比べると高い。

適応させるために、正解ラベル付きデータ（教師付きデータ）を追加的に用いてモデルの学習を進める手法である。同手法では、追加学習させたいタスク（転移学習でいう目標ドメイン）に関連するラベル付きデータを作成する手間が必要となる。

4.2.4 RLHF：Reinforcement Learning from Human Feedback

強化学習（Reinforcement Learning）は、エージェントが環境との相互作用を通じて報酬を最大化する行動方針を学習する方法である。強化学習は LLM でも採用されており、RLHF（Reinforcement Learning from Human Feedback）が典型例である。まず、人間のアノテーター（何らかのラベルを付ける人）が LLM の出力を評価し、その評価に基づいて報酬モデルを訓練する。次に、この報酬モデルを用いて、LLM が新たに生成する出力を評価し、強化学習アルゴリズム（例えば、Proximal Policy Optimization; PPO）を通じて LLM をチューニングする。こうした手法により、LLM が人間（評価者）の好みに合致する応答を生成する能力を高めることができる。人間のフィードバックを強化学習に介在させるため、Human Feedback という名前が付いている。LLM は事前学習で得た情報をもとに新しい状況やタスクに評価関数を通じて適応していくため、RLHF は転移学習の一形態とみなされる[32]。GPT-3.5 の開発においては、図表 25 の下段にあるシリーズ（InstructGPT）で RLHF が採用され、GPT-3 からの能力改善を果たしており、これが ChatGPT のチャット能力を高めた。Anthropic の Claude や DeepMind の Gopher においても RLHF が活用されている。

32：松井・熊谷（2024）は 13 章で強化学習一般における転移学習を解説している。

4.2.5 ICL：In-Context Learning

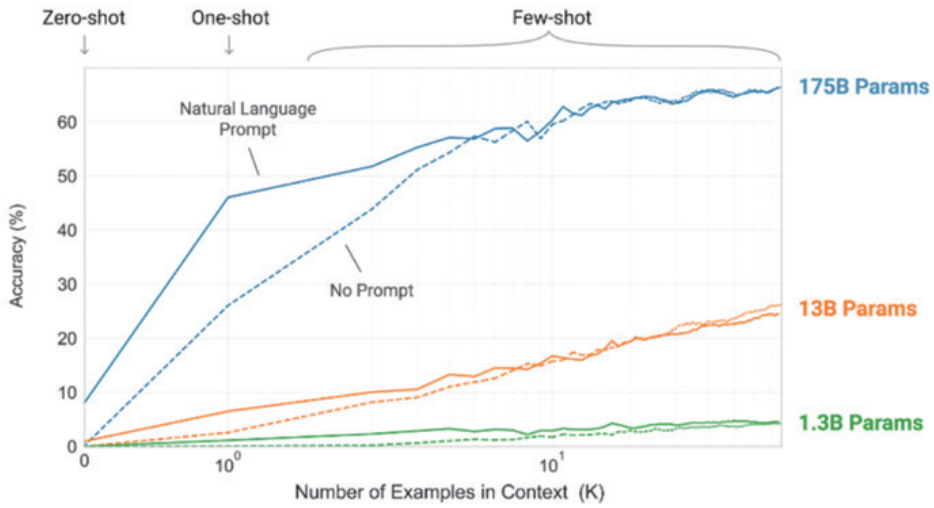
ファインチューニングや追加学習では、LLM のパラメータ再設定が行われるが、LLM のパラメータを不変としつつ転移学習を活用するコスト効率の高い手法が考案された。現在のプロンプトエンジニアリングに繋がる In-Context Learning（ICL）である。ICL は GPT-3 の開発において発見された。具体的には、LLM の推論時に、インプット指示（プロンプト）の補足情報として望ましい回答事例や回答様式を示すことで、優れた回答を引き出す手法である。これにより、事前学習された知識を新しい状況やタスクに適用することが可能となる。

ICL は、文脈に沿った指示を新しいデータとして与え、LLM が特定のタスクを遂行する能力を高める手法であり、転移学習が新しいタスクに対してモデルを適応させるメカニズムと類似している。また、ICL は、モデルが事前に学習した一般的な知識を活用し、特定のタスクに対して柔軟に適応することを可能にする。例えば、少量のデータで LLM に高性能を発揮させ、迅速に新しいタスクに適応させることができるという利点がある（追加学習が不要）。このようにプロンプトに含まれる少ない情報だけで LLM がタスクを遂行できることは、転移学習の持つ適応能力を反映したものである。

ICL では、事例を一つだけ与えるケースを One-shot learning（あるいは One-shot プロンプト、以下同じ）と呼び、数個の場合を Few-shot learning

と呼ぶ。ここから派生して、回答事例なしでプロンプトを与えるのを Zero-shot learning と呼ぶようになった。プロンプトエンジニアリングでは、ICL だけではなく、LLM に回答者としての立場（役割）を与えたり、特定の口調やスタイルを指示したりすることも含まれる。関連する背景情報や文脈をプロンプトに含めることで、より正確で関連性の高い応答を生成できるようになる。例えば、過去の会話履歴や特定の知識ドメインに関する情報をプロンプトに追加する事例があげられる。図表 30 は、GPT-3 において発見された ICL の性能改善を数値検証した研究内容を示している。モデル規模ごとに ICL の効果が表れていることや（小規模モデルでも右上がりとなっており、規模差なりの改善を示している）、大規模モデルでは One-shot learning だけで大きな性能改善効果があることが観察されている。

図表30　GPT-3でのOne-shot/Few-shotでの性能改善



出所）Brown et al.（2023）

4.2.6 マルチモーダル

転移学習は LLM のマルチモーダル化でも活用されている。多くのマルチモーダルモデルは、既存の単一モーダル（例えば文章のみ）で学習されたモデルをもとにしており、その知識を利用して他のモーダル（例えば、画像や音声）を理解・生成する能力を追加する。例えば、OpenAI の CLIP（Contrastive Language-Image Pre-training）は、大規模なテキストデータセットで事前学習された言語モデルと、画像データセットで事前学習された視覚モデルを組み合わせており、テキストと画像のペアを用いて学習することで両者の関係を理解する能力を獲得している。

転移学習を用いることで、テキストと画像など異なるモーダル間の知識を統合し、それぞれのモーダルが持つタスク実行能力を相互に補完することができる。例えば、DALL-E や Flamingo のような生成 AI モデルは、テキストから画像を生成する、あるいは画像を理解してテキスト表現する能力を有している。

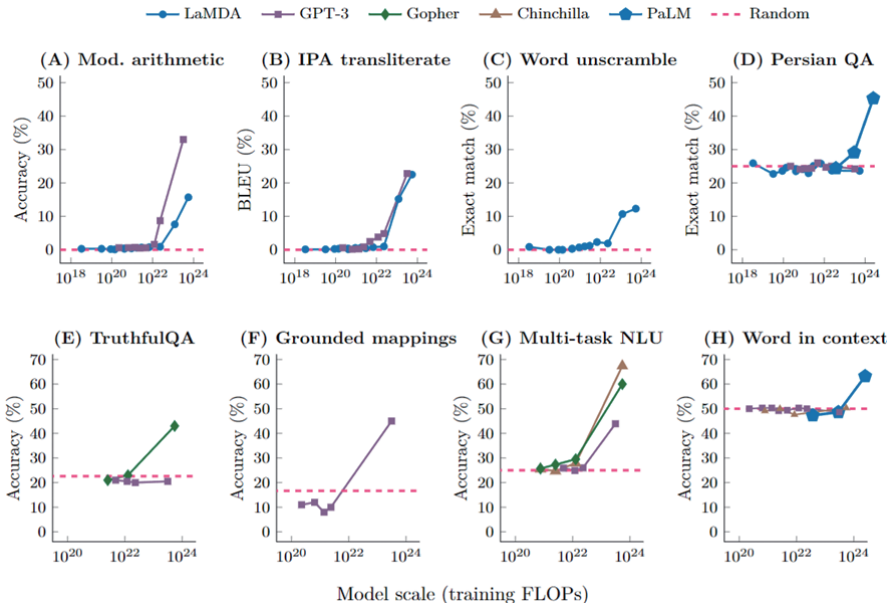
このように元々は文書の生成 AI であった LLM に対して、画像や音声の生成 AI の知識やスキルを転移統合することが行われており、LLM は文章生成 AI を超えて高い汎用性を持つ生成 AI 化してきている。このため、生成 AI と LLM の区分は曖昧化しつつある。

4.3 創発現象

LLM の大規模化に伴って発見された点に、創発現象（emergent ability）がある。これは、LLM の文脈では、モデルが小規模な間は対応することができなかった問題やタスクが、規模がある水準を超えると突然解けるようになる（能力が出現する）ことを指す。前述の In-context learning がその一例である。Wei et al.（2022）は他の様々な創発事象を取り上げている。図表 31（A）では、3 桁の加算と減算、2 桁の乗算をテストする算術ベンチマークの検証結果が示されている。横軸に示されたモデル規模がある閾値を超えると突然 LLM が計算能力を獲得することがわかる。発音記号から単語をスペルアウトする問題（B）、単語内でアルファベットの順番をランダムに入れ替えたものから元の単語を復元する問題（C）、ペルシャ語の質問応答（D）、質問に対する真偽の判定（E）、方位などの概念を扱う能力（F）、数学、歴史、法律など多様なトピックをカバーするテスト（G）でも、同様な創発現象が観察されている。

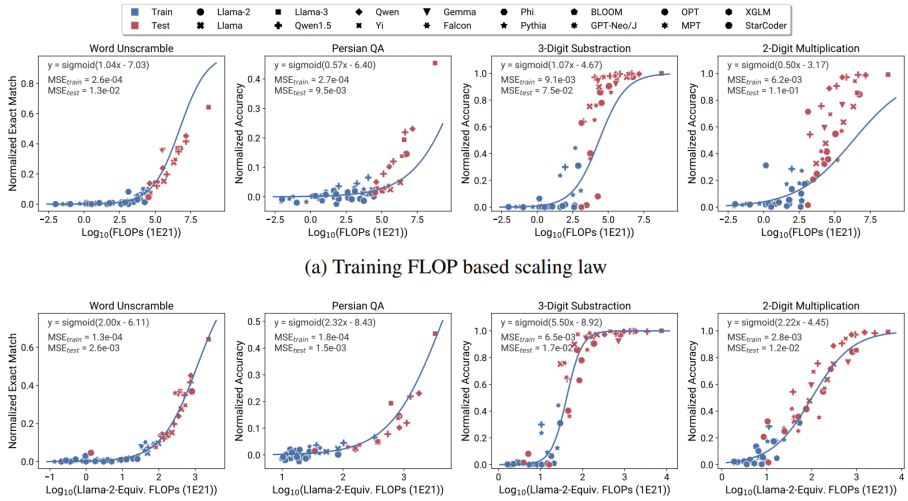
LLM の様々なタスクに関する能力の非連続的な変化は、モデルやベンチマークテストの開発に不確実性をもたらし、開発方針や評価を困難なものとする（べき乗則の逆ケース）。これらの能力が本当に非連続的であるのか、使用される評価指標の癖によるものなのか、あるいは検証の際の解像度が不足しているせいなのか、議論や検証がなされている。Ruan, Maddison and Hashimoto（2024）は、解像度を上げて詳細に見れば突然変化しているのではなく徐々に能力は上がっていることを指摘している（図表 32）。このほか、Schaeffer, Miranda and Koyejo（2024）は、評価指標の設計や選択によって突然能力が上昇しているように見えているのであり、指標を変えると連続的な変化になりうることを指摘している。ただし、今できないことが、どこまで大規模化したらできるようになるかという不確実性の問題は引き続き解決されていない。

図表31　創発現象の発生



出所）Wei et al.（2022）

図表32　創発発生地点の高精度計測



出所）Ruan, Maddison and Hashimoto（2024）

5．RAGとFine-tuning、AIエージェント

5.1 汎用LLMの限界と３つの対応法

４節でみてきたように、LLM は高性能化を続けている。しかし、企業内での利用や顧客情報を活用したサービス提供に生成 AI/LLM を利用する場合、内部情報や秘匿すべき情報をどのように活用するかという問題に直面する。そうした情報は「汎用」LLM の学習情報セットには含まれていない（含めてもいけない）。また、汎用 LLM の学習には膨大な計算資源が必要となり、ユー

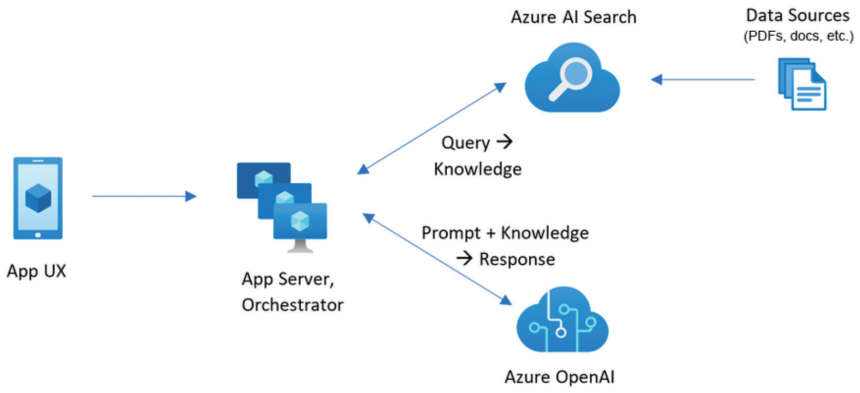
ザ企業にとって追加学習を行うことは容易ではない。開発企業ですら学習内容の頻繁なアップデイトは行われておらず、通常、カットオフデイトが設定され、その時点までで得られた情報に基づく学習が行われている。更には身の回りの細かい粒度の情報や頻繁に更新される情報を LLM に学習させるのは著しく非効率的であり、そのようなことは行われていない。それゆえ、今日の東京の天気や社内規定や、近所にあるお勧めレストランを聞いても汎用 LLM は正しく回答できず、ハルシネーションを起こすか、情報がないと回答するだけである。

こうした問題に対して、①非公開情報 / 秘匿情報を LLM の外部にデータベースとして持たせる、②コスト効率的なファインチューニングによりこれらの情報を追加学習させる、③一般公開情報を Web サイトで検索したり、特定情報を提供するサービスサイトから取得してくる、というアプローチが考案された。5 節ではこれらを順に説明する。

5.2 RAG

情報を外部データベースに保有させるアプローチとして、RAG（Retrieval Augmented Generation、検索拡張生成）という手法が Lewis et al.（2020）によって提唱された。①利用対象となりうるような情報をデータベースとして LLM の外部に構築し、② LLM への問いかけに対して関連性が最も高い情報をデータベースから抽出する。③これを LLM への問いかけ情報とセットにして、その情報の範囲内で回答するようなプロンプト指示と合わせて LLM に送る、というシステムである。図表 33 はその構成の一例である。これにより、汎用 LLM が一般情報に基づいた回答を返してしまいハルシネーションや不適切な回答が生成される可能性を抑制することができる。同時に、非公開情報 / 秘匿情報を活用したサービスを情報にアクセスする権利がある適格者だけに提供することが、システム上の工夫によって可能となる。RAG においては、LLM は、知識のストレージとアウトプット機能というより、与えられた情報に基づいてプロンプトの要求に従って回答を作成する自然言語生成機、すなわち言葉を操れる便利な UI として機能している。

図表33　RAGのシステムアーキテクチャの事例



出所）Azure website

RAG システムの構築においては LLM 以外の IT 知識が必要となる。LLM の性能以上に、情報データベースを的確に構築し、そこから回答生成に必要な情報を不足なく検索・収集してくる検索システムの性能が鍵となる。情報検索においては、ベクトル化データベースを構築し、問い合わせ情報を分散表現したうえで、類似度が高い（内積が大きい）ものをベクトル化データベースから選択する方法が LLM の自然な延長線上で発展してきた。しかし、ベクトル化技術 / 分散表現技術にこだわる必要は必ずしもなく、伝統的な検索技術も応用可能である。データベースをインデックス化し、検索用の指標、例えば TF-IDF（特定の文書における特定の単語の重要性を評価するための指標）やその改良版の BM25 を用いて、適切な情報をインデックス検索する手法である[33]。Microsoft 社のクラウドサービス Azure では、検索サービスとして Azure AI Search（図表 33）を提供しているが、ベクトル検索とインデックス検索に加えてセマンティック検索の 3 つをハイブリッドで用いて検索精度を改善させている。検索精度の向上は RAG のパフォーマンスに直結する。

33：RAG の隆盛により以前より発展してきた検索技術に再び注目が集まっている。検索技術の詳細は、例えば打田他（2022）に最新技術まで含めた解説がある。

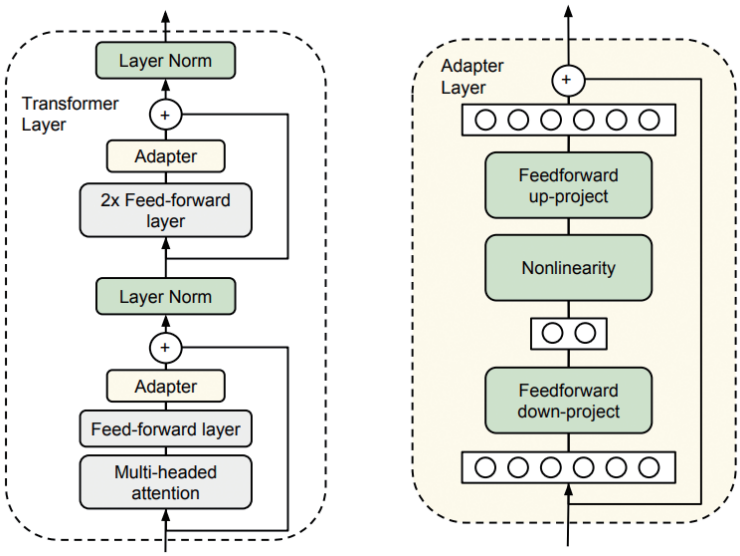
2023 年には、企業が顧客向け、社内向けに生成 AI サービスを構築する動きが加速したが、RAG は、2024 年上期においても主要な開発手法となっている。これは後述する PEFT が、効率化されているとはいえ Fine-tuning で LLM のパラメータを再推計するという大がかりかつ精度を引き上げるのが容易ではない手法であるのに対し、RAG においては情報データベースの構築と検索精度が確保できれば、LLM 自体をエンジニアリングする必要がないからである。そのため、RAG 構築を支援する様々なサービスが提供されており、6 節では簡単な事例を用いて複数の RAG 構築例を紹介する。

5.3 PEFT

2018 年に GPT-1/BERT が事前学習と Fine-tuning の組み合わせを提唱したが、その後の LLM の大規模化加速で Fine-tuning のコストがより大きなものとなり、開発企業でもない限り現実的な選択肢ではなくなった。そこで、事前学習で獲得されたパラメータは固定したうえで、モデルの一部追加と当該部分のパラメータを調整することで Fine-tuning を実行しようというアプローチが登場した。これらの技法は PEFT（Parameter Efficient Fine-Tuning）と呼ばれる。

Houlsby et al.（2019）はアダプター層（adapter layers）と呼ばれる追加モジュールを既存の事前学習モデルに挿入し、アダプター層のみをファインチューニングする手法を提案した。実際に BERT と複数の学習データセットで改善度合いを検証し、同アプローチが有効なことを確認している。アダプター層はフィードフォワード型の NN に類似した構造となっており、図表 34（右図）のように中間層の次元を圧縮することでパラメータ効率性を高めている。

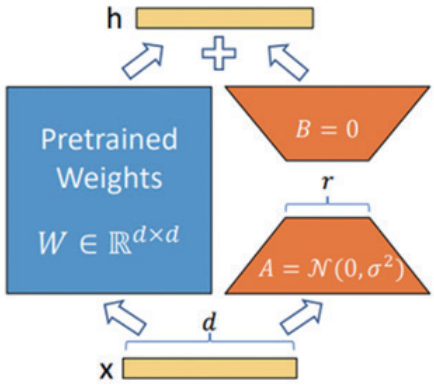
図表34　アダプター層の追加と内部構造の例



出所）Houlsby et al.（2019）

Hu et al.（2021）は、Transformer モデルの Attention 機構の QKV 生成行列や、その後のフィードフォワード NN 層の重み行列に対して、行列の次元を圧縮することでパラメータ数を削減する手法を考案した。行列のランクを落とす手法のため、LoRA（Low-Rank Adaptation）と呼ばれる。その際、事前学習されたモデルのパラメータは固定し、それと並列する形で次元圧縮されたモデルを並列に追加することにより、事前学習で習得した学習内容を劣化させることなく、新規情報を学習させている（図表 35）。

図表35　LoRAの概念図：左がオリジナルモデル、右が並列設置した次元圧縮モデル



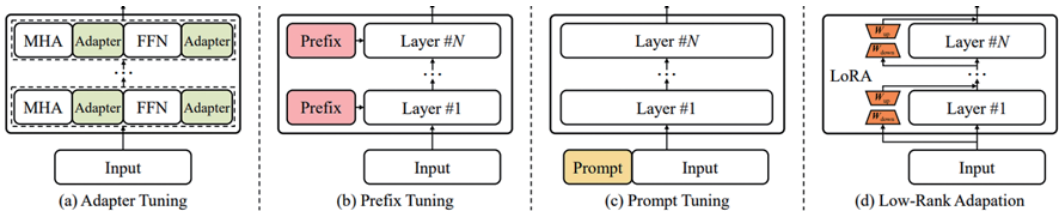
出所）Hu et al.（2021）

Li and Liang（2021）は、Prefix-Tuning と呼ばれる手法を提案した。図表 36（b）がその概念図である。事前学習済みモデルに対する入力値に特定のプレフィックス（接頭語 / 接頭情報）を追加し、そのプレフィックスに対するパラメータ部分のみを学習する（他の入力データにかかるパラメータは不変）。このほか、Selective と呼ばれる手法があり、Transformer の各層にあ

るバイアス（定数項）のみを再学習する手法である。

なお、PEFT といえども、GPT-3 以降のようにモデルが巨大化すると新規に学習するパラメータが増加する。その次元を絞り込み過ぎるとパラメータが過少なことによる性能低下が生じかねない（図表 37 の左下がり部分）。そこで、図表 36（c）にあるような Prompt Tuning も考えられた。プロンプトエンジニアリングと似ているが、タスクに適したプロンプトを自動的に生成・最適化する点が異なる。前者は人間が試行錯誤しながらプロンプトを工夫するアプローチであり、使い方のチューニングに相当する。

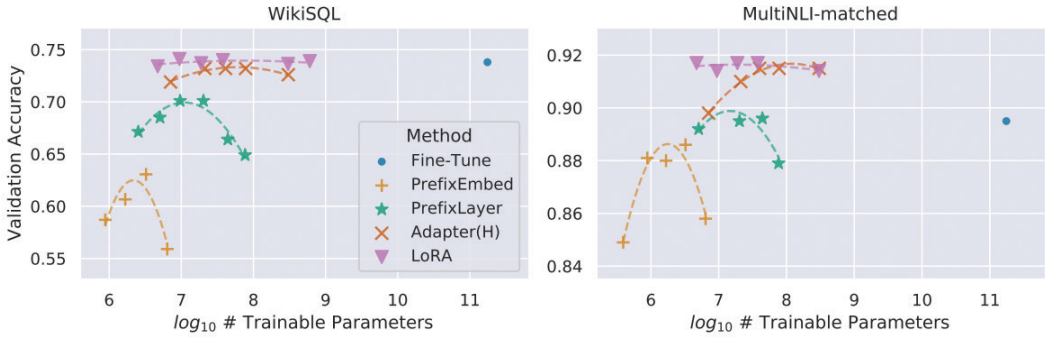
図表36　ファインチューニングのアーキテクチャ



出所）Zhao et al.（2023）

図表 37 は、LoRA を提唱した研究論文が示した PEFT 各手法の比較である。オリジナルモデルをフルにファインチューニングした結果が右側の丸印で（パラメータ数が多い）、各手法についてパラメータ数を変更しながら性能の変化を見たものが各手法別に点線で示されている。LoRA の学習性能の高さと、パラメータを絞り込んでも劣化しにくい性質が観察されている。

図表37　PEFT各手法の性能比較



出所）Hu et al.（2021）

5.4 RAGとPEFTの選択

図表 38 に RAG と PEFT の特徴を対比した。適切な応用分野としては、定型的にはチャットボットや FAQ、ニュース作成、専門分野の診断（医療 / 法 / IT）、内部ナレッジ集積・活用、専門分野翻訳・要約など共通するところが多いが、定性的にみるとサービスの内容に応じて両者を使い分けた方がよいことがわかる。

図表38　RAGとPEFTの比較

	RAG	PEFT
長所	外部知識の活用により広範な情報資源にアクセス可能、情報の更新頻度も操作可能、モデルサイズが小さくとも性能が期待できる	事前学習済みのモデルを効果的に活用、フルパラメータの再学習に比べ破滅的忘却のリスクが低い
短所	知識データベースのメンテナンスが必要、パフォーマンスが検索精度に依存、検索した情報を LLM が上手くまとめきれないことや一般情報のノイズが入る場合もある	適応できるタスクに制限がある場合がある、モデルの対応領域に限界がある
実装	外部知識データベースの構築や、データパイプラインの実装知識が必要	各種 PEFT 固有のモデル追加（アダプター層の追加等）が必要
拡張性	知識ベースの拡張で柔軟に可能	新しいタスク（ドメイン）ごとにファインチューニングが必要
適切な応用分野	広範な知識を必要とし、知識の更新頻度も比較的高い分野	ファインチューニング・追加学習が行いやすい分野、知識領域の限定と低頻度更改

出所）筆者作成

5.5 AIエージェントの活用、LangChainフレームワーク

RAG、PEFT に続く 3 番目の対応は、AI エージェントの活用である。今日の東京の天気をリアルタイムで知りたい場合、天気予報サイトに API 接続して情報を収集し、これをもとに LLM で文章化するほうが確実である。ここでも LLM は自然言語インターフェイスとして機能しており、知識や情報収集は API 先を訪問して情報を収集してくる AI エージェントが担うことになる。

こうしたシステムの構築においては、RAG と同様、LLM は構成要素の一部に過ぎず、AI エージェントを機能させるセンターや、収集情報の受け渡しを担うデータパイプラインの設計が必要となる。以上のような IT システム構築のコストを削減してくれる生成 AI フレームワーク兼ライブラリとして LangChain が 2022 年 10 月にリリースされ、2023 年を通じて急速に普及した。LangChain は、LLM を活用したアプリケーションの開発を支援するためのフレームワークであり、これを活用することで、データ取得、前処理、モデルの呼び出し、出力解析など、複数のステップを統合して一連のワークフローを構築できる。ライブラリとして様々なサービスを備えており、ベクトル化データベースの検索や、エージェントの起動など、種々のタスクに対応するためのモジュールを提供し、開発者がチェーン（処理の流れ）を短時間のうちに構築することを可能にする。直感的な API を提供しており、特定の機能やタスクを呼び出すためのインターフェイスが利用できる。6 節では Python から LangChain や LlamaIndex を使って RAG や AI エージェントを構築した簡単な事例を紹介する。

6．アプリケーション実装の進化と学び方

企業が生成 AI を活用した自社サービスを構築する際には、適切な LLM の選択以外にも種々の課題が伴う。どのような IT 環境で、どのような（LLM 以外の）サービスと連携させて開発するのか、社内情報など非公開情報を利用する場合には安全な環境をどう構築するのか、非公開情報の活用は RAG と PEFT のいずれを使うのか、利用者のアクセスをどう管理するのか（誰が何を知っていいのか / 情報を使っていいのか）、品質管理にかかる問題への対応（例えば情報の更新やサービスシステムのメンテナンス）、サービスの内容や技術の陳腐化に伴う寿命予測（サービス構築の投資判断）、生成 AI サービスの公正性や公平性、倫理の確保、データ保護規制などの法令順守、開発スタイル（内製化か外注か、人材の確保は）といった種々の課題が伴う。更に高い次元の課題としては、企業全体のデジタル経営戦略に生成 AI をどう位置付けて、資金や人材といった経営資源を確保・投入するのか、組織体制をどうデザインするのかといった経営上の論点もある。

これらの課題のうち、本節ではアプリケーション実装にかかる IT インフラの選択や、実装を進めるに際しての学びのプロセスの参考になるような情報を提供する。これらは、IT 戦略すなわち経営戦略と直結する論点となるため、技術論に止まる話ではない。

6.1 段階的学びのステップを通じて知るインフラ技術

学びという視点から、生成 AI アプリケーションサービスの使い方と作り方を順に並べると、例えば以下のような 6 ステップが考えられる。段階的学びの過程において生成 AI アプリケーションサービス構築に必要な IT インフラ知識も高まっていく。

Step1　生成AIアプリケーションサービスの体験

まず、商用（有償 / 無償）の公開サービスを利用して、生成 AI サービスにどのような種類のものがあり、どのような UI で提供され、どのような UX（ユーザー体験）をもたらしているのかを「身をもって知る」ことが第一歩目である。例えば、ChatGPT や Gemini（のチャットサービス[34]）、1 節で紹介した notebookLM や GPTs の各種サービスを利用し、研究者であれば Elicit という論文検索・内容分析ツールや perplexity（出典に基づいて回答を作成する対話型検索エンジン）を使ってみることがあげられる。現在の ChatGPT はエージェント機能がバックエンドで動いており、専門的な質問をするとインターネットで検索した結果を情報源として GPT-4 の言語生成能力と汎用知識をあわせて回答文を作成している。また、notebookLM は RAG の実装事例であり、本稿を読んだ後に利用すると UI の背後で動いている仕組みがイメージできる。notebookLM の裏で動いている LLM は Google の Gemini 1.5 Pro である。これと同様に、各種の生成 AI アプリケーションサービスでは、LLM 自体は前面に出てこず、その機能を使って何をユーザが行いたいか（サービスプロバイダが何を届けたいか）に焦点をあてた作り込み方になっている。

34：GPT シリーズはチャット web アプリケーションサービスを ChatGPT と別名シリーズで提供しているが、Gemini シリーズでは、API サービスを提供する LLM ライン（Gemini 1.5 Pro/Flash など）とチャット web アプリケーションサービスを同名としている。

以上のような生成 AI 入門学習体験だけでも、ここ 1 ～ 2 年の LLM 関連技術の大幅な進歩が体感できる。ChatGPT が初登場してきたころとはサービスの内容や質が劇的に変化している点に未だに気が付いていないビジネスパースンや研究者は少なくない。最新のサービスを使ってみるという最初の一歩が一番重要である。

Step２　Python入門

生成 AI アプリケーションサービスの開発に必要なコンピュータ言語として Python を学ぶ。JavaScript/TypeScript を Node.js 実行環境で動かすアプリケーションを作成する方法に対応している LLM や関連サービスもあるが、Python が最も汎用性が高く、かつ AI / 機械学習全般に活用されているため応用範囲も広い。また、学びにかかる初期コストも比較的小さい。Python を用いると LLM ベンダーなどから提供されている API 接続サービスが簡単に実現できる。curl コマンドを用いてシェルから投げる方法もあるが、学びには不便である。Python の利用環境はインタラクティブなノートブック形式でコードを実行できる Jupyter Notebook や同様なスタイルでクラウドの高性能計算環境を間借りできる Google Colaboratory（Google Colab）がある。このほか、VSCode（Visual Studio Code）のような汎用コードエディタに Python 拡張機能をインストールして使用する方法もある。最初からクラウド上でのアプリケーション構築に進むなら、Azure Notebooks（Jupyter Notebook）環境や AWS の SageMaker 環境を用いる方法もある。

Step3　APIを使ったLLM使用

Python の基本的な使い方が理解できたら、LLM の公式資料や、ネット上にある無料学習教材、各種の体験記事（Qiita や note、Zenn に大量にある）などを参考にしながら、API を経由して商用 LLM サービスを使ってみる。商用 LLM 開発企業は API を通じて LLM サービスを提供している。Step1 で示した生成 AI アプリケーションサービスは UI を各社で作り込んではいるが、その入力情報は高性能の商用 LLM（サービス内容によっては閉鎖環境下のクラウド LLM）に対して API で引き渡され、処理されたものを受け取ってブラウザで提示するといったサービス構成を基本的に採用している。顧客向けサービスを提供する各社が LLM を各々の自社サーバで運用しているわけではない（そうしたケースも存在しうるが、何らかの経済合理性などがある場合に限られるであろう）。

このため、生成 AI アプリケーションサービスでは API の利用が必須となる。その利用法は使いやすく整備されており、アカウントの取得や API-Key の管理などに気を付ければ API 接続は容易に行うことができる。Python などを使って自ら作成した生成 AI アプリケーションサービスから API を通じて外部 LLM を呼び出すことが可能となる。

Step4　LangChain/LlamaIndex/Difyを使う

汎用 LLM を単体で使うだけでは、企業内情報や顧客情報を活用した

サービスは提供できない。外部データを活用するRAGをLangChainやLlamaIndexを用いて作成してみる。これらはオープンソースとして提供されているため、誰でも利用することができる。LangChainやLlamaIndexの公式ドキュメント、ネット上の解説マテリアル、書籍（布留川 2023・2024、田村 2023）などを参照すると簡単に学ぶことができる。LLMの使い方は学べても、アプリケーションサービスに仕立てるには多くの周辺技術の学びと実装が必要となるが、LangChain/LlamaIndexはそのコストを大きく引き下げた。これらは生成AIアプリケーションサービス開発の民主化の象徴的存在となっている。これらを用いるとAIエージェント系のサービスも容易に実装できる。その際、ネット検索に利用するサービスでAPI登録、API-Key発行が必要となるが、簡単に取得できる。

また、ノーコードで生成AIアプリケーションサービスを構築する技術も進歩している。Difyはオープンソースの開発プラットホームで、2024年に入って急速に注目を集めるようになった。クリックだけでRAGシステムほか多様な生成AIアプリケーションサービスを構築することができる。LangChain/LlamaIndexは、便利とはいえコードを書ける技術を必要とした。Difyを始めとして、今後も増えるであろうノーコード開発ツール群は、生成AIの民主化を更に推し進めることになろう。

Step5　クラウドサービスの基本を学ぶ

Step4だけでは、非公開情報を閉鎖環境で運用し、利用権のあるユーザだけに対してサービスを提供することができない。これを実現するITインフラとして、自社サーバにオープンソースのLLMを立てるという選択肢もありえるが、クラウドサービスを使ったセキュアな閉鎖環境やユーザの登録・認証システムの構築が、開発速度や運用の柔軟性、コストの面で現実的である。クラウドベンダー各社は、LLMだけでなくアプリケーションサービスを構築する際に必要となる様々なマネッジドサービスを提供している。これらを用いることで、情報データベースや検索サービス、データパイプライン、UIとなるWebアプリケーションサービスなどを一体活用できる利点がある（図表33参照）。Amazon Bedrockのように様々な最新のLLMを選択可能なかたちで提供するクラウドサービスもある。もちろん、LangChainのようなオープンソースサービスをクラウド上で利用することも考えられるが、本番環境に展開していった際の可用性や統合運用性を考えると、クラウドのマネッジドサービスの利点は大きく、構築したいシステム内容やコスト面も合わせての判断となろう。

Step6　Azure OpenAIやAmazon Bedrock、Google Cloud Vertex AIを使う

パブリッククラウドサービス活用の基本がマスターできると、Azure OpenAIやAmazon Bedrock、Google Cloud Vertex AIなどの統合サービスを用いた生成AIアプリケーションサービスに到達する。ここに至って、課題であった①企業内部情報や顧客情報をどう活用して生成AIアプリケー

ションサービスを構築するか、②その際、情報管理の安全性をどのようにして確保するかが解決可能となる。クラウドベンダーのサイトには、システム構築に必要な情報や学習マテリアルが多く掲載されている。永田他（2024）や御田・熊田・森田（2024）、坂本（2022）ほかクラウドサービス上での生成 AI アプリケーションサービス構築に関する書籍も出版されている。

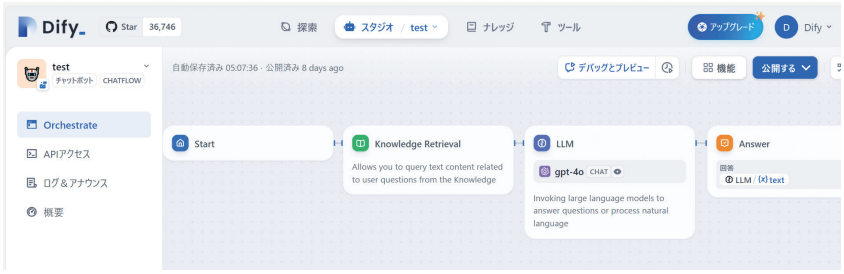
6 つのステップを順に踏む必要はないが、一足飛びにクラウドサービスを契約して開発を進めようとすると、技術的スタックを内部で積み上げていないため、外注に頼りがちとなる。また、PoC での手触り感もないまま進めることになる。生成 AI に限らず、技術進歩が非常に早く、サービスの陳腐化リスクが高い分野では、外注依存のサービス開発体制はマイナス面が大きい。ここに示したのは学び方のステップの一例に過ぎないが、サービスを創造・運営していくビジネスの現場が何らかの形で内製化に関与していく経営に切り替えていく必要がある。これは生成 AI に限らず、IT サービス産業化するデジタル社会でのビジネス全般に共通するテーマである。

6.2 RAG実装の３事例

1 節で利用例として示した notebookLM は RAG を一般向け公開 Web アプリケーションサービスとして実装したものであった。図表 3 のアプリケーション画面左側は、読み込ませたレポート 24 本がリストされており、これらの情報はベクトルデータベース化されている。下の入力欄からの問い合わせに対応して、データベース情報が検索され、適切なものが抜き出され、プロンプトとあわせて Gemini 1.5 Pro に API 経由で送信され、回答を受け取ってブラウザに表示されるという情報処理の流れとなる。

これと同じものをノーコード開発の Dify で簡単に作成することができる。図表 39 に示した Dify のパネルには、問い合わせを受ける「スタート」から右側に向かって「知識データベース」、「LLM（GPT-4o、筆者所有アカウントの API-Key を利用）」、「回答」という構成が取られていることが示されている。図表 40 は情報データベースの中身である。URL 指定で登録されたレポートが並んでいる（図表 3 と同じ）。図表 41 はその中身の一つを見たもので、レポートが細かい部分に分割されて（チャンクされて）格納されていることがわかる。文章はチャンク単位でベクトル化されて保管され、利用時に検索対象となり、対象となった場合はプロンプトと合わせて LLM に送られる。関連する部分を過不足なく選択するため、文章は細かくチャンクされる。図中に示されている情報は、元は html ファイルであるが、パーサ（parser）機能によって不要なタグ情報を削除して格納されている。Dify は Firecrawl という高機能の Web クローラー兼 Markdown 形式変換 API サービスを今年 6 月に組み込んだため、図表 41 のように綺麗な整形物を簡単に得ることができる。URL の下位ディレクトリ指定や排除もメニューから簡便に行える。元情報が PDF であった場合、段落・見出し・図表などを識別し、適切にパーサする能力が重要となる。こうしたパーサの性能も RAG を支える重要な構成要素となっている。

図表39　Difyのメインパネル



出所）筆者作成

図表40　Difyのナレッジパネル

出所）筆者作成

図表41　htmlファイルからの格納状態:チャンク分割

出所）筆者作成

最後に、Python を用いて LangChain を使った RAG システム構築の事例を紹介する。図表 42 はその一部を示しており、まず Tavily という AI エージェント専用に構築された検索エンジンを使って、ブロックチェーンを使った資産のトークン化に関する質問に関連した情報をインターネット上から収集してくるよう AI エージェントに命令している。具体的には、RWA（Real World Asset）のトークナイゼーションに際して日本のどのような法律が関

係してくるかを問うている。図中には収集した結果を示しており、ネット上にあった異なる pdf ファイルから、関連度が高そうな箇所をパーサして整えたうえで、収集していることがわかる。

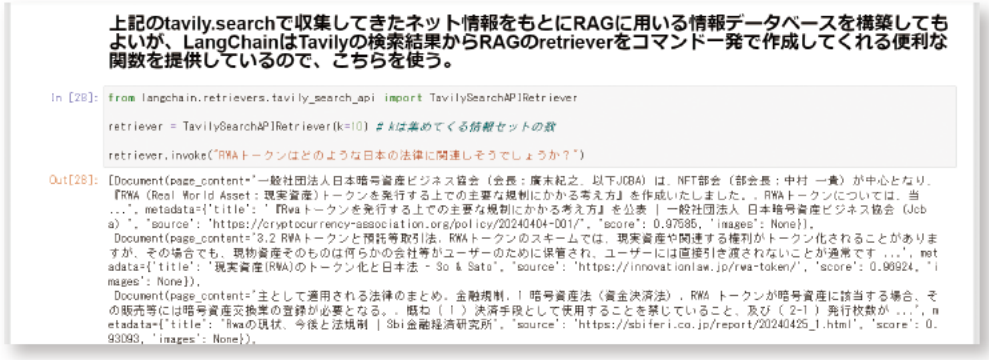
図表42　Webサーチ用エージェント

出所）筆者作成

図表 43 では、収集した情報から情報データベースを作って検索装置を繋ぐという標準的な手続きは行わず、一気に RAG を構築する手法を採用している。TavilySearchAPIRetriever で RAG オブジェクトを作成し、そこに web サーチの結果をストアするコードを示している。これがライブラリの読み出しと生成オブジェクトの利用を含めて僅か 3 行で完結する。3 行目では、同オブジェクトの呼び出し（invoke）機能から質問内容を読み込んで、回答を生成している。図表 44 では、この情報データベースからの検索結果を LLM（ここでは GPT-4o のチャットシステム）に与えて、文章を生成させたものを表示している。プロンプトでは、提供された情報のみに基づいて回答するよう指示を与えている。これも 10 行程度のコードで実現できている。

このように LangChain を利用すると、簡単に情報データ指定型の RAG や web サーチ・エージェント型の RAG を構築することができる。また、PEFT で LoRA を試してみたい場合は、HuggingFace という AI 技術のライブラリから Python に読み込むことができる。この場合は計算資源を必要とするので、ローカル PC の Python 環境ではなく Google Colab を利用した方がよい。

図表43　サーチ結果からのRAGオブジェクト生成とその使用



出所）筆者作成

図表44　RAGを使ったチャットシステム

出所）筆者作成

7．おわりに

本稿では、生成 AI ウォークスルーと題して、ニューラル言語モデルの基礎、現在の LLM に繋がるモデル発展、様々な LLM の群雄割拠、発展の過程で発見されてきた多様な転移学習の新形態、それらがもたらした利用法の拡大、RAG や PEFT の技術、実装に必要な技術群とその学び方、実際の RAG 構築事例を紹介した。こうした情報は、生成 AI 技術を金融実務に活用していく際の学びや実践の水先案内（Pilot）となろう。6 節で紹介した Step1 からの実践を通じて、生成 AI のビジネス活用が決して難しいものではないことが体感できると思う。また、筆者が参加した証券アナリスト協会主催の座談会（和泉他 2024）では、金融機関が生成 AI 活用を推進する際の様々な課題が議論されており、こちらも参考となろう。

この分野は、過去数年のうちに急速に発展したため、ほぼ全員が新規参入者として同じスタートラインにいる。もちろん過去の技術スタックの有無はロ

ケットスタートの実現に影響するが、技術の民主化によりハードルは大きく引き下げられた。6 節で紹介したノーコード開発が典型例である。全力で走り出したものから市場や付加価値（利益）の創造に近づいていけることは間違いない。本稿がその Pilot となれば幸甚である。

最後に、自然言語処理技術の可能性について 1 点触れたい。SBI 金融経済研究所の所報 5 号で水田（2024）は、人工的な金融市場を用いたシミュレーションの可能性について様々な研究を紹介している。人工市場の研究においては、エージェントの振る舞いをどう現実的に設定するか、どこまで複雑な振る舞いをモデル化できるかが、結果の妥当性やリアリティを担保する鍵となる。生成 AI は膨大なコーパスとそこに含意された人間の思考パターンを巨大なモデルとパラメータ群で学習している。これを人工市場のエージェントとして用いることには大きな可能性があると考えられる。

類似の発想が Ha and Schmidhuber（2018）によって World Models として提唱されており、強化学習やロボティクス、AI シミュレーションの分野で大きな影響を与えている[35]。World Models では、エージェントが環境の内部モデル（ワールドモデル）を構築し、観察からの学習によって「ワールド」の状態や状態の推移則を認知する。そのうえで、潜在的な報酬を最大化するように自らの行動を決定する。最適化のスコープのバリエーションは色々ありえて、例えば、エージェントが行動を履行することがワールドへ何がしかの影響を及ぼし、これが自己へフィードバックしてくる点も考慮したうえで行動選択を決定するという設定も考えられる。

同論文では、仮想空間上での車の自動運転学習が取り上げられているが、モダンマクロ経済学の DSGE モデルを学んだものは、発想の親和性の高さと DSGE の限界を意識するであろう。すなわち、エージェントである代表的個人は完全に動学最適化を解いたうえで行動を選択する（未来にわたる動学的パスを含めて決定し、毎期生じる外生ショックに対して最適化計算をやり直す）という合理的個人の仮定とその限界である。この仮定を緩めるとアドホックなモデルがいくらでも構築でき、観測された現実が仮定によって自在に説明できてしまう。上述した人工市場の研究におけるエージェントの振る舞い設定の適切さに関わる難しさと同根の問題である。

しかし、人間の認知や意思決定の複雑さにかかる部分が重要であることは、近年の非伝統的金融政策の振り返りで話題に取り上げられている「デフレのノルム」の議論においても確認される。財サービス価格を引き上げると市場シェアを失うため、生産コストの抑制、とりわけ賃金の抑制で対応しよう（幸いデフレであり雇用者との賃金交渉もそれを可能にしているほか、労働者にとってもデフレならば賃金据え置きは容認可能）といった「ワールド（世界の在りよう）」の認知が企業側にも労働者側にも成立し、すなわちデフレのノルム（社会通念）が定着し、自己実現的な均衡に陥ったというものである。こうした均衡が成立し、長期間継続してしまった理由を検討し、どのような外生ショックやメカニズムがデフレ均衡からの離脱をもたらすのかという検証を行う際に、ピュアな DSGE モデルの適用には限界がある。

経済環境や金融市場というワールドを人間がどう認知し、それを行動に反

35：David Ha は、元 Google Brain で現在は Sakana AI の CEO である。Jürgen Schmidhuber は、本稿で紹介した LSTM の考案者の一人である。ちなみに、Sakana AI の CTO である Llion Jones は、Transformer モデルの考案チームのメンバーである。

映させるかを考える際に、World Models の考え方は一つのヒントになろう。金田・坂地（2023）は、気候変動の原因と結果に関するナラティブがどのように成立したか、株式市場がこうしたナラティブにどう反応したかを BERT やテキストからの因果抽出技法によって検証している。認知科学と自然言語処理と金融経済学の接点であろう。こうしたワールドモデルが繋ぐ学際的アプローチは、Web3 がイメージする分散分権型の仮想社会で、エージェントがどう振る舞い、その集合体としてのシステムがどう振る舞うかを検証するうえでも有益なツールとなりそうである[36]。LLM をはじめとする AI 技術の応用範囲は広い。

36：OpenAI が 2024 年 2 月に公表した Sora は、テキストプロンプトから動画を自動生成する生成 AI モデルとして注目を集めた。黒いレザージャケットとサングラスを身につけた女性がアジア風の夜の繁華街の濡れた道路を歩く有名な動画である。メディアでは動画生成 AI として取り上げられたが、OpenAI の website は、"Video generation models as world simulators" と紹介しており、サイトの URL もそうネーミングされている。Sora は世界モデルの構築技術の一つであり、OpenAI は「Sora は AGI を達成するための重要なマイルストーン」と述べている。

参考文献

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2014), "Neural machine translation by jointly learning to align and translate," arXiv:1409.0473, 2014.

Bengio, Yoshua, Réjean Ducharme, Pascal Vincent and Christian Jauvin (2003), "A neural probabilistic language model," *Journal of Machine Learning Research*, Vol.3, pp.1137-55, February 2003.

Bengio, Yoshua, Réjean Ducharme and Pascal Vincent (2000), "A neural probabilistic language model," *Advances in neural information processing systems*, vol.13, 2000.

Bojanowski, Piotr, Edouard Grave, Armand Joulin, and Tomas Mikolov, "Enriching word vectors with subword Information," arXiv:1607.04606, 15 July 2016.

Brown, Tom, et al. (2020), "Language models are few-shot learners," Advances in neural information processing systems 33 (NeurIPS 2022), pp.1877-1901, 2020.

Cho, Kyunghyun, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk and Yoshua Bengio (2014a), "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," arXiv:1406.1078, 2014.

Cho, Kyunghyun, Bart van Merriënboer, Dzmitry Bahdanau and Yoshua Bengio (2014b), "On the Properties of Neural Machine Translation: Encoder–Decoder Approaches," arXiv:1409.1259, 2014.

Clark, Kevin, Urvashi Khandelwal, Omer Levy and Christopher D. Manninget (2019), "What does BERT look at? An analysis of BERT' s attention," arXiv:1906.04341, 2019.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee and Kristina Toutanova (2018), "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," arXiv:1810.04805(v1), 11 October 2018.

Elman, L. Jeffrey, (1990), "Finding structure in time," *Cognitive Science* vol.14, 2, pp.179-211. March 1990.

Feng, Junxi, et al. (2019), "Reconstruction of porous media from extremely limited information using conditional generative adversarial networks," *PHYSICAL REVIEW E* 100, 2019

Gunasekar, Suriya et al. (2023), "Textbooks Are All You Need," arXiv: 2306.11644(v2), 2 October 2023.

Ha, David and Jürgen Schmidhuber (2018), "World Models," arXiv: 1803.10122(v4), 9 May 2018.

Harsoor, Sharan (2023), "Transformer model and variants of transformer (ChatGPT)," Medium, 7 June 2023. https://pub.aimind.so/transformer-model-and-variants-of-

transformer-chatgpt-3d423676e29c
Hendrycks, Dan, et al. (2021), “Measuring massive multitask language understanding.” arXiv:2009.03300(v3), 12 January 2021. (Note this is not the version1 in 2020.)
Hochreiter, Sepp, and Jürgen Schmidhuber (1997), “Long short-term memory,” *Neural computation 9.8*, pp.1735-80, 1997.
Hoffman, Jordan, et al. (2022), “Training Compute-Optimal Large Language Models,” arXiv: 2203.15556(v1), 29 March 2022.
Hopfield, John (1982), “Neural networks and physical systems with emergent collective computational abilities,” *Proceedings of the national academy of sciences 79.8* pp.2554-58, 1982.
Houlsby, Neil, et al. (2019), “Parameter-efficient transfer learning for NLP.” International conference on machine learning, PMLR, arXiv:1902.00751(v2), 13 Jun 2019.
Hu, Edward J., et al. (2021), “Lora: Low-rank adaptation of large language models.” arXiv:2106.09685(v1), 2021.
Jordan, I. Michael, “Serial order: A parallel distributed processing approach,” Institute for Cognitive Science, University of California, San Diego, ICS Report 1986, May 1986.
Kaplan, Jared, et al. (2020), “Scaling laws for neural language models.” arXiv:2001.08361, 23 January 2020.
Karpathy, Andrej (2015), "The unreasonable effectiveness of recurrent neural networks," Andrej Karpathy Blog, posted on May 21, 2015.
Le, V. Quoc and Tomas Mikolov, “Distributed representations of sentences and documents,” arXiv:1405.4053, 16 May 2014.
Li, Xiang Lisa and Percy Liang (2021), “Prefix-Tuning: Optimizing Continuous Prompts for Generation,” arXiv:2101.00190(v1), 1 Jan 2021.
Patrick Lewis et al. (2020), “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” arXiv: 2005.11401(v1), 22 May 2020.
Minaee, Shervin, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain and Jianfeng Gao (2024), “Large language models: A survey.” arXiv:2402.06196, 2024.
Mikolov, Tomáš, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur, “Recurrent neural network based language model,” *Interspeech* 2010, pp.1045-48, 26 September 2010.
Open AI et al. (2023), “GPT-4 Technical Report” , arXiv:2303.08774(v1), 15 March 2023. (Latest version is 6th.)
Penedo, Guilherme, et al. (2023), “The RefinedWeb dataset for Falcon LLM: Outperforming curated corpora with web data only,” *Advances in Neural Information Processing Systems* 36, 2023.
Peters, E. Matthew, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee and Luke Zettlemoyer (2018), “Deep contextualized word representations,” arXiv:1802.05365, 15 February 2018.
Radford, Alec, Karthik Narasimhan, Tim Salimans and Ilya Sutskever (2018), “Improving language understanding by generative pre-training,” 2018.
Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei and Ilya Sutskever (2019), “Language models are unsupervised multitask learners.” OpenAI blog 1(8),9, 2019.
Rothman, Denis (2021), Transformers for Natural Language Processing, Packt Publishing, 2021.（デニス・ロスマン（2022）、『Transformer による自然言語処理』、黒川利明訳、朝倉書店）

Ruan, Yangjun, Chris J. Maddison and Tatsunori Hashimoto (2024), “Observational scaling laws and the predictability of language model performance.” arXiv:2405.10938(v1), 17 May 2024.
Sardana, Nikhil and Jonathan Frankle (2023), “Beyond Chinchilla-optimal: Accounting for inference in language model scaling laws,” arXiv:2401.00448(v1) 31 Dec 2023.
Schaeffer, Rylan, Brando Miranda, and Sanmi Koyejo (2024), “Are emergent abilities of large language models a mirage?,” *Advances in Neural Information Processing Systems* 36, 2024.
Sutskever, Ilya, Oriol Vinyals and Quoc V. Le, “Sequence to Sequence Learning with Neural Networks,” Advances in neural information processing systems, 2014.
Thompson, D. Alan (2024), “Chinchilla data-optimal scaling laws: In plain English,” LifeArchitect.ai, February 2023, updated Jun 2024. https://lifearchitect.ai/chinchilla/
Vasnetsov, Andrey (2024), “BM42: New Baseline for Hybrid Search,” Qdrant web site, 1 July 2024, https://qdrant.tech/articles/bm42/
Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin, “Attention is All you Need,” *31st Conference on Neural Information Processing Systems*, NIPS 2017.
Wei, Jason et al.(2022), “Emergent abilities of large language models,” arXiv:2206.07682(v2), 26 October 2022.
Yang, Jingfeng, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Shaochen Zhong, Bing Yin and Xia Huet (2024), “Harnessing the power of LMs in practice: A survey on ChatGPT and beyond,” *ACM Transactions on Knowledge Discovery from Data* 18.6, pp.1-32, 2024.
Zhao, Wayne Xin, et al. (2023), “A survey of large language models,” arXiv :2303.18223 2023.
Zheng, Huiting, Jiabin Yuan and Long Chen, “Short-Term Load Forecasting Using EMD-LSTM Neural Networks with a Xgboost Algorithm for Feature Importance Evaluation,” *Energies*, 10(8), 8 August 2017.
和泉潔他（2024）、「生成 AI が変える金融市場・資産運用業界への影響（座談会）」、証券アナリストジャーナル 62 巻第 2 号、pp.54-78、日本証券アナリスト協会、2024 年 2 月
今泉允聡（2021）、『深層学習の原理に迫る：数学の挑戦』、岩波科学ライブラリー 303、岩波書店
Weights and Biases Japan (2024)、「Nejumi LLM リーダーボード 3 開発の経緯とその評価から見えてきたこと」、note、2024 年 6 月 30 日、https://note.com/wandb_jp/n/nd4e54c2020ce
打田智子・古澤智裕・大谷純・加藤涼・鈴木翔吾・河野普策（2022）、『検索システム：実務者のための開発改善ガイドブック』、ラムダノート
宇根正志、「機械学習による予測・推論の公平性：金融サービスにおいて求められる配慮とは」、金融研究、第 43 巻第 1 号、日本銀行金融研究所、2024 年 1 月
岡崎直観・荒瀬由紀・鈴木潤・鶴岡慶雅・宮尾祐介（2022）、『自然言語処理の基礎』、オーム社
岡崎直観他、「Swallow コーパス：日本語大規模ウェブコーパス」、言語処理学会第 30 回年次大会発表論文集、2024 年 3 月
御田稔・熊田寛・森田和明（2024）、『Amazon Bedrock 生成 AI アプリ開発入門』、SB クリエイティブ、2024 年 6 月
金田規靖・坂地泰紀（2023）、「BERT と因果抽出を用いた気候変動ナラティブの可視化・指数化」、日本銀行金融研究所、ディスカッションペーパーシリーズ 2023-J-7、2023 年 6 月
神嶌敏弘編・人工知能学会監修（2015）、『深層学習』、近代科学社
斎藤康毅（2018）、『ゼロから作る Deep Learning ② 自然言語処理編』、オライリー・ジャパン
坂本俊之（2022）、『Vertex AI で作る AI パイプライン入門』、C & R 研究所、2022 年 11 月
ストックマーク株式会社編（2021）、『BERT による自然言語処理入門：Transformers を使った

実践プログラミング』、オーム社
副島豊（2024）、「金融システムの未来像を探る中央銀行の挑戦」、SBI 金融経済研究所、所報第５号、2024 年３月
――― （1996）、「ニューラルネットワークアプローチによる経済分析：モデルの概要と金融政策への応用例」、金融研究、第 15 巻第３号、日本銀行金融研究所、1996 年８月
田村悠（2023）、『LangChain 完全入門：生成 AI アプリケーション開発がはかどる大規模言語モデルの操り方』、インプレス社
坪井裕太・海野裕也・鈴木潤（2017）、『深層学習による自然言語処理』、機械学習プロフェッショナルシリーズ、講談社
永田祥平他（2024）、『Azure OpenAI ではじめる ChatGPT/LLM システム構築入門』、技術評論社
布留川英一（2023）、『OpenAI GPT-4/ChatGPT/LangChain 人工知能プログラミング実践入門』、ボーンデジタル
布留川英一（2024）、『Google Gemini 1.5 ／ LlamaIndex ／ LangChain 人工知能プログラミング実践入門』、ボーンデジタル
松井孝太・熊谷亘（2024）、『転移学習』、機械学習プロフェッショナルシリーズ、講談社
水田孝信（2024）、「人工市場：金融市場のコンピュータ・シミュレーション」、SBI 金融経済研究所、所報第５号、2024 年３月