第 1 章

近年の自然言語処理技術における各種手法の概要と特徴

（株）アナリティクスデザインラボ　野守 耕爾

はじめに

2022 年 11 月にアメリカの OpenAI によってリリースされた AI チャットボットサービス“ChatGPT”をきっかけに世界中で生成 AI ブームが起こることとなった。その賢すぎる AI の突然の誕生に人類は大いに興奮し，技術革命とも呼べるその可能性の大きさから，巨大 IT 企業はこぞって生成 AI 技術の開発を加速させ，競争は激化した。特にその技術の中心となったのが生成 AI を実現する大規模言語モデル（Large Language Model, LLM）である。これはインターネット上に人類がこれまで蓄積し続けてきたテキストデータを大量に学習させた汎用的な言語モデルであるが，そのコアとなるのが深層学習を使った Transformer という自然言語処理技術である。Transformer は 2017 年に Google とトロント大学の研究者によって発表されたが，いわばこの手法が自然言語処理領域において革命を起こしたと言っても過言ではない。またその Transformer にいたるまでもさまざまな自然言語処理技術が開発され続けてきたわけだが，昨今の生成 AI ブームもこれまでの研究者たちによって紡がれたあくなき探究心の結晶といえる。本節ではそんな近年の自然言語処理技術について，その全体像を整理しながら俯瞰した理解を進めるべく，代表的な手法を取り上げ，その概要と特徴を解説する。

1.　近年の自然言語処理技術の整理

自然言語処理（Natural Language Processing, NLP）とは，人間が使う言葉を機械で処理できるようにする技術の集合体であるが，そのためにはその言葉を機械が認識できる数値の形式で表現する必要がある。その代表的な方法が，文書や単語をベクトルで表現するというものである。その最も基本的な技術に「形態素解析」があり，これは文書に含まれる単語を分解するという手法である。たとえばその単語の出現頻度でもって文書をベクトル表現できる。また，その単語の出現頻度をベースとした自然言語処理技術に「トピックモデル」という手法がある。これは主に 1990 年代から 2000 年代初頭に開発された技術であり，単語の出現頻度の教師なし学習によって，文書に潜むトピックを抽出するというものである。このアウトプットを通じて，大量の文書でも人間がその概要を理解することの助けとなる。一方，主に 2010 年代以降では「深層学習モデル」を使った自然言語処理技術が数多く登場した。いわゆる第 3 次 AI ブームの中で生まれた技術であるが，特に 2017 年に発表された Transformer が大きなブレイクスルーとなり，2018 年以降では ChatGPT に代表される大規模言語モデルの開発が盛んとなった。こうした深層学習による教師あり学習を用いた自然言語処理技術は，文書を翻訳したり要約したり，文書の変換や生成を目的としたものとなる。

本節では近年の自然言語処理技術として，形態素解析を基礎とした「トピックモデル」と「深層学習モデル」を取り上げる。これらの技術領域における代表的な手法を整理すると図 1 のようにまとめることができる。トピックモデルは人間が文書を理解するための技術であるのに対して，深層学習モデルは機械が文書を理解するための技術と解釈できる。各種手法の詳細な説明は専門書に委ねるが，本節では各種手法の概要と特徴について簡単に解説し，技術の全体像を俯瞰して理解する。

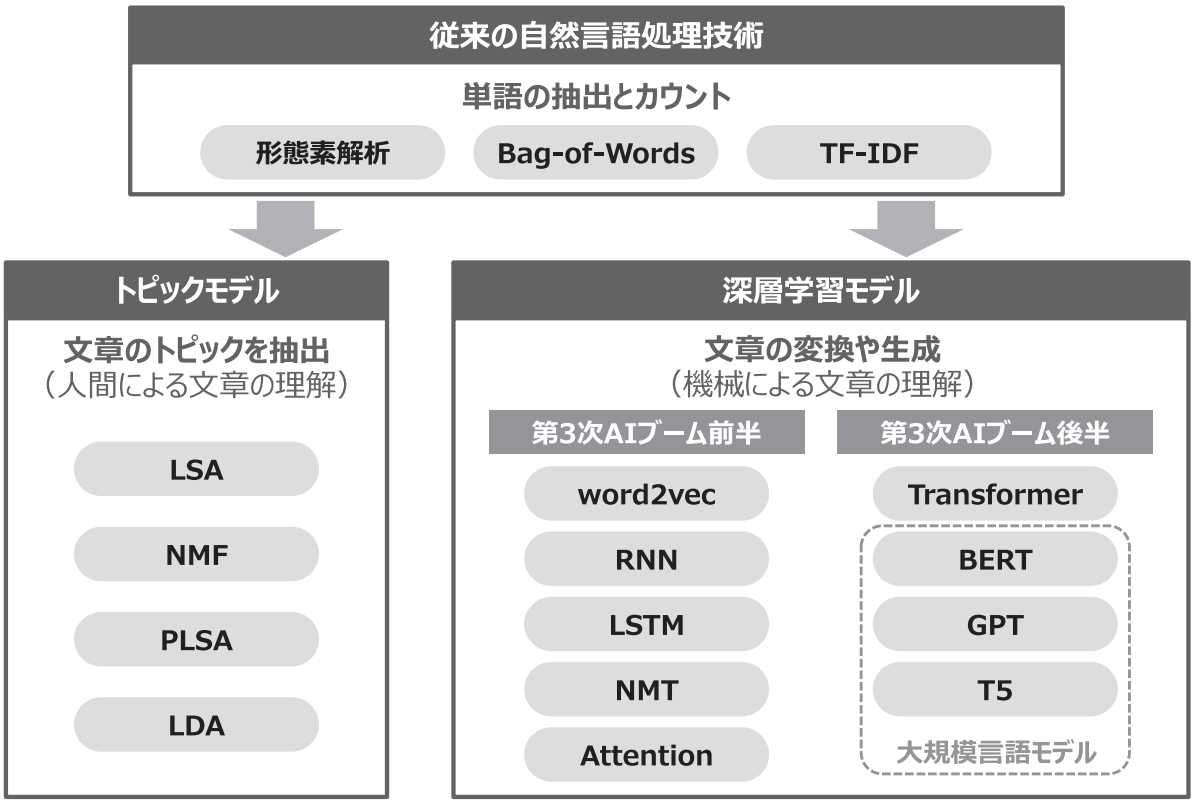


図 1　近年の自然言語処理技術の整理

2.　従来の自然言語処理技術

従来の自然言語処理技術として，ここでは「形態素解析」「Bag-of-Words」「TF-IDF」を取り上げる。文書に含まれる単語を分解・抽出し，その単語の出現頻度でもって文書をシンプルにベクトル表現する基本的な処理技術である。テキストデータの分析といえば，テキストマイニングが有名であり，ビジネスの場面でも広く利用されている。テキストマイニングの説明は 5.2 にて改めて後述するが，そのテキストマイニングでアウトプットされる可視化や統計解析のベースとなっているのがこれらの処理技術である。ここでは“従来の技術”と括っているが，今日でも広く利用されている技術であり，自然言語処理技術において基盤を提供している大変重要な技術である。

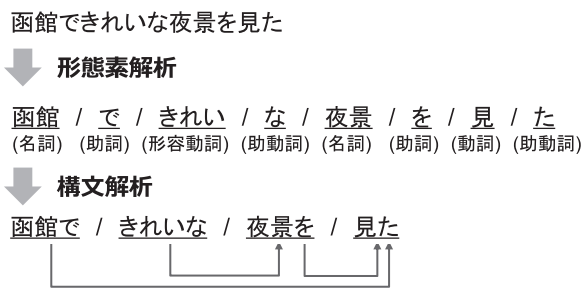
2.1　形態素解析

形態素解析とは，文章を単語に分割し，その単語の品詞を割り当てる技術である [1)]。正確には単語ではなく，形態素と呼ばれる意味を持つ最小単位の文字列として分割する手法だが，実用面では，形態素ではなく単語の単位で分割されることが一般的である。補足として，単語とは一つ以上の形態素から構成される言語の単位である。単語と形態素の違いの例として，たとえば「観光地」という単語は「観光」「地」という 2 つの形態素で，「宿泊施設」という単語は「宿泊」「施設」という 2 つの形態素で構成される。このように，日本語の意味情報を抽出するには最小単位の形態素よりも単語の方が適しているとされる。なお，英語の文章の場合は単語がスペースで区切られているため，日本語よりも形態素解析がしやすい言語といわれる。

形態素解析の発展した技術として構文解析というものもある。構文解析とは，文法規則に基づいて文章の構造を解析し，単語間の関係性を識別する技術である。たとえば，主語と述語や，修飾語と被修飾語といった係り受け関係を抽出する。日本語の場合は単語ではなく文節を単位に係り受け関係を抽出するのが一般的である。文節とは，日本語を意味のわかる単位に区切ったものであり，助詞や助動詞は名詞や動詞とセットで括られる。なお，日本の小学校では，文節は「〜ね」で区切るものと教わることが多い。

形態素解析と構文解析の例を図２に示す。これらの処理により，文章に含まれる単語とその品詞，また単語間（文節間）の係り受け関係を抽出することができる。なお，実際に抽出される単語や係り受けは，登録されている辞書によってそれぞれ原形に変換された形で抽出されることが一般的である。

形態素解析と構文解析には公開されたフリープログラムがあり，形態素解析では JUMAN，ChaSen，MeCab などが，構文解析では KNP や CaboCha などが広く知られている。また，テキストマイニングと呼ばれるツールは，この形態素解析と構文解析の２つを基本技術としていることが一般的であり，これによって処理された情報を使ってさまざまな形で可視化や統計解析をするツールとなる。



形態素解析で抽出される単語と品詞

単語	単語の原形	品詞
函館	函館	名詞
で	で	助詞
きれい	きれい	形容動詞
な	だ	助動詞
夜景	夜景	名詞
を	を	助詞
見	見る	動詞
た	た	助動詞

構文解析で抽出される係り受け

函館⇒夜景
きれい⇒夜景
夜景⇒見る

図２　形態素解析と構文解析の例

2.2　Bag-of-Words

Bag-of-Words（BoW）は，最もシンプルに文書をベクトル化する手法である。形態素解析によって文書に含まれる文書を抽出し，各文書にどの単語が何回出現したのかをカウントする手法であり，その出現頻度という数値によって文書一つひとつをベクトルとして表現している。Bag-of-Words のベクトル表現は文書×単語という行列形式のデータで考えると分かりやすい。一つの行に一件の文書を，一つの列に一つの単語を取ったとき，各セルの値はその文書内におけるその単語の出現回数を示す。Bag-of-Words によるデータ形式の例を図３に示す。これは最もシンプルな文書のベクトル表現だが，この処理だけでもテキストデータのさまざまな定量分析が可能であり，とても強力な処理手段である。テキストマイニングのツールで実行される可視化や統計解析は基本的にこのデータ形式をベースにしている。

文書ID	ホテルの口コミコメント	部屋	広い	快適	空調	良い	効く	綺麗	清掃	風呂	駅	近い	便利	スタッフ	対応	丁寧	朝食	美味しい	レストラン	…
1	部屋が広く快適で、部屋の空調も良く効きました。	2	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	
2	部屋は綺麗に清掃されていて、お風呂も快適でした。	1	0	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	
3	駅の近くで便利で良く、部屋も広くて良かったです。	1	1	0	0	2	0	0	0	0	1	1	1	0	0	0	0	0	0	
4	スタッフの対応が良く、部屋も丁寧に清掃されていました。	1	0	0	0	1	0	0	1	0	0	0	0	1	1	1	0	0	0	
5	朝食が美味しく、レストランも広くて綺麗で良かったです。	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	1	1	1	
…	……																			

図３　Bag-of-Words の例

一方，シンプルが故の課題も存在する。たとえば，Bag-of-Words ではすべての単語は同等に扱われ，その単語が全文書で見たときにどれくらい重要であるかは認識されない。単語の位置や順序, 使われ方, 文脈における意味といった情報も保持しておらず，その文字列の出現頻度のみの情報となっている。また，単語の種類の数がベクトルの次元数となるため，高次元なデータとなって複雑で，計算処理が難しくなるという課題もある。

2.3 TF-IDF

Bag-of-Words において，単語の重要度が考慮されていないという課題に対応した手法に TF-IDF がある。TF-IDF は，各文書における単語の重要度を数値化する処理であり，TF という指標と IDF という指標の積によって計算される。

TF（Term Frequency）は，各文書の中における単語の頻度を示す値であり，頻度が多い単語ほど重要であると判断する。なお，文書の長さの影響を受けるため，その文書で出現する全単語頻度で除して正規化することが多い。一方，IDF（Inverse Document Frequency）は，ある文書で頻出する単語でも，それがどの文書でも頻出するなら重要とはいえないと判断する指標である。一般的には，log（全文書数 ／ 単語 w が出現する文書数）で定義され，多くの文書で現れる単語は値が小さくなり，特定の文書にしか現れない単語は値が大きくなるように調整される。Bag-of-Words のデータから TF-IDF の処理をする例を図 4 に示す。

TF-IDF は，テキストデータの分析でよく用いられる前処理であり，たとえば文書間の類似度の分析では，TF-IDF の処理をしたベクトルを使い，その cos 類似度を計算することが多い。

Bag-of-Words

ID	部屋	広い	快適	空調	良い	合計頻度
1	2	1	1	1	1	6
2	1	0	1	0	0	2
3	1	1	0	0	2	4
4	1	0	0	0	1	2

TF-IDF

ID	部屋	広い	快適	空調	良い
1	0	0.116	0.116	0.231	0.048
2	0	0	0.347	0	0
3	0	0.173	0	0	0.144
4	0	0	0	0	0.144

（例）ID=3の「良い」のTF-IDF

TF = 2 / 4 = 0.5
IDF = log (4 / 3) = 0.288
TF-IDF = 0.5 * 0.288 = 0.144

図 4　TF-IDF の例

3.　トピックモデル

Bag-of-Words による「文書×単語」の行列データをインプットに，教師なし学習で文書に潜むトピックをアウトプットする，「トピックモデル」と呼ばれる技術の各種手法について解説する。従来のテキストマイニングでは，形態素解析で抽出された単語を単位に可視化や統計解析を実行し，文書全体の傾向を把握しようとするが，トピックモデルでは，そうした単語群をいくつかのトピックに集約する。つまり，単語単位ではなく集約されたトピックを単位に文書全体の傾向を把握できる手法となる。特に大規模なテキストデータの場合，従来のテキストマイニングでは膨大な単語が抽出されるため，その単語群をベースにした可視化はとても複雑になり，解釈性が薄れてしまう。解釈性を向上させるために，Bag-of-Words のデータに対してクラスター分析で文書をグルーピングすることもあるが，なかなか良い結果が得られにくいことが多い。その理由として，Bag-of-Words は列となる単語の数が膨大でかなり高次元なデータとなるため，距離に基づいた通常のクラスター分析を実行してしまうと，データ間の距離がどれも非常に離れ，妥当な結果が得られにくくなる次元の呪いという問題が生じてしまう。これに対してトピックモデルでは，距離に基づかないでこの高次元データの次元数を削減し，膨大な単語群をトピックに集約することができる。これに

より人間が文書をシンプルかつ効率的に理解するのに役立つ。トピックモデルは主に 1990 年代から 2000 年代初頭にかけて開発が盛んとなった技術であるが，ここではその代表的な手法として「LSA」「NMF」「PLSA」「LDA」を取り上げる。

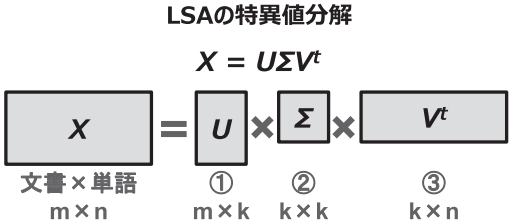
3.1 LSA

LSA（Latent Semantic Analysis，潜在意味解析）は，Bag-of-Words による「文書×単語」の行列を SVD（Singular Value Decomposition, 特異値分解）によって次元削減することでトピックを抽出する手法である。1990 年に発表された手法であるが [2]，その基礎となる考え方の研究はそれ以前から存在している。情報検索の分野では LSI（Latent Semantic Indexing）とも呼ばれている。

LSA による特異値分解のイメージを図 5 に示す。LSA の具体的な処理は,「文書×単語」行列を①「文書×トピック」（左特異ベクトル），②「トピック×トピック」（特異値），③「トピック×単語」（右特異ベクトル）の 3 つの行列に分解することである。特異値「トピック×トピック」は対角行列であり，その対角成分には特異値が大きい順に配列される。左特異ベクトルの「文書×トピック」と右特異ベクトルの「トピック×単語」は, トピックを示すベクトル（左特異ベクトルでは各列，右特異ベクトルでは各行）が直交しており，各トピックの軸は数学的に互いに独立している。SVD による数学的な処理は，元の行列を最もよく近似する行列に分解すること，つまり元の行列と分解後の行列の誤差を最小化する最適化問題を解くことである。特に，SVD はこの最適解の存在が数学的に保証されているという点で強力であり，計算過程のシンプルさから計算効率も高いメリットもある。

SVD に類似する手法に PCA（Principal Component Analysis, 主成分分析）があり，PCA は SVD を用いた分析手法と捉えることができるが，両者の関係について簡単に説明する。PCA は元のデータセットの分散を最大化させる方向（主成分）を見つけ出し，最も情報量の多い特徴を抽出することを目指している。その主成分の方向に元のデータを射影することで，元のデータの重要な情報を保持しながら次元を削減する。SVD は分散を最大化させるという目的で用いられるものではないが，PCA を効率的に実行できる手法として用いられるため，PCA は特定の条件下における SVD の応用の一例と見ることができる。つまり，SVD は PCA よりも一般的な手法といえるが，実用面でいえば両者は同様の手法のように扱われることがある。

SVD は大きな値に引っ張られてトピックが抽出される傾向があるため，LSA を実行する際には，Bag-of-Words に TF-IDF などで重み付けした行列を用いることが多い。また LSA の課題として，分解する行列の要素に負の値を許容しているため，結果の解釈が難しくなることや，結果が学習データに完全に依存するため，過学習を起こしやすく，新しい文書のトピックは推定できないことなどが挙げられる。



LSAでアウトプットされる各行列の例

①文書×トピック

U	トピック1	トピック2	トピック3	トピック4
文書1	-0.47	-0.75	-0.46	0.11
文書2	-0.61	-0.10	0.69	-0.37
文書3	-0.27	0.41	-0.54	-0.68
文書4	-0.58	0.52	-0.10	0.62

②トピック×トピック

Σ	トピック1	トピック2	トピック3	トピック4
トピック1	7.7	0	0	0
トピック2	0	2.8	0	0
トピック3	0	0	2.0	0
トピック4	0	0	0	0.6

③トピック×単語

V^t	単語1	単語2	単語3	単語4	単語5
トピック1	-0.51	-0.49	-0.38	-0.37	-0.47
トピック2	-0.27	0.14	0.80	-0.50	-0.09
トピック3	0.47	0.43	-0.35	-0.67	-0.14
トピック4	0.61	-0.74	0.23	-0.18	0.07

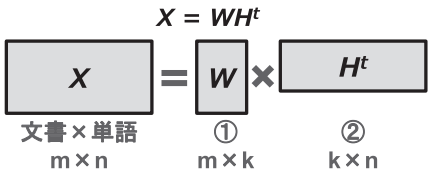
図 5　LSA の特異値分解

3.2 NMF

NMF（Non-negative Matrix Factorization，非負行列因子分解）は，Bag-of-Words の「文書×単語」の行列を，2つの非負の行列「文書×トピック」と「トピック×単語」に分解することでトピックを抽出する手法で，1999 年に発表された [3]。NMF による行列分解のイメージを図 6 に示す。

計算アルゴリズムは，元の行列と分解後の行列の積との誤差を最小化することを目的関数に，初期値を与えた反復計算により最適解を得る。誤差の定義の仕方は，平方ユークリッド距離や KL ダイバージェンスなどが使われる。反復計算は乗法更新式（Multiplicative update rule）が代表的であり，これは得られる行列が必ず非負であるという制約条件化で適用される勾配降下法である。LSA と比較して，分解後の行列の要素がすべて非負であるため，結果の解釈がしやすいメリットがある。

LSA と異なり，分解された「文書×トピック」と「トピック×単語」の行列は，各トピックのベクトル間で直交しておらず，各トピックは数学的に互いに独立していない。これにより，トピック間に意味的な重複が生じる可能性がある。また，LSA と同様に，NMF でも結果が学習データに完全に依存するため，過学習を起こしやすく，新しい文書のトピックは推定できないという課題がある。



NMFでアウトプットされる各行列の例

①文書×トピック

W	トピック1	トピック2	トピック3	トピック4
文書1	0.22	0.48	0.16	0.39
文書2	0.35	0.12	0.07	0.88
文書3	0.11	0.58	0.24	0.14
文書4	0.29	0.27	0.60	0.20

②トピック×単語

H^t	単語1	単語2	単語3	単語4	単語5
トピック1	4.34	0.03	1.21	2.24	1.87
トピック2	3.41	1.89	2.81	0.46	1.03
トピック3	0.38	2.24	1.09	5.62	0.06
トピック4	1.85	3.66	0.08	2.11	4.16

図 6　NMF の行列分解

3.3 PLSA

PLSA（Probabilistic Latent Semantic Analysis，確率的潜在意味解析）は，確率モデルを導入してトピックを抽出する手法である。特異値分解によってトピックを抽出する LSA の処理を確率的な枠組みによって発展させたもので，1999 年に発表された [4]。情報検索の分野では PLSI（Probabilistic Latent Semantic Indexing）と呼ばれることもある。

PLSA では，Bag-of-Words の「文書×単語」の行列を確率的に分解することでトピックを抽出する。PLSA の確率モデルのイメージを図 7 に示す。PLSA のモデルは文書 d における単語 w の出現確率 P（w|d）を，潜在的なトピック z を介してモデル化した Aspect モデルを起点としている（Aspect とは文書内に潜むトピックを指す）。Aspect モデルを式展開することで，文書 d と単語 w の同時確率 P（d, w）を，3 つの確率分布① P（d|z），② P（w|z），③ P（z）に分解して表現する。この 3 つの確率分布を Bag-of-Words の「文書×単語」行列データに基づき推定する。具体的には，P（d, w）の対数尤度関数を最大化する EM アルゴリズムを実行し，初期値を与えた反復計算により最適解を得る。なお，PLSA は NMF とは異なる手法であり，NMF は確率的な仮定を置いたモデルではないが，KL ダイバージェンスを用いた NMF と PLSA が，同じ目的関数を最適化していると解釈できることが示されている [5]。

PLSA のメリットとして，LSA では「文書×単語」の行列に対して事前に TF-IDF などで重みづけする必要があったが，PLSA では確率的な処理によりそうした重みづけをせずに実行することができる。また，PLSA は推定される P（d|z）

とP（w|z）によって，文書および単語のトピックに対する関連度が所属確率として出力されるため，結果が解釈しやすいメリットがある。なお，各トピックは互いに独立しているという数学的な仮定を置いている。

PLSAのデメリットとしては，入力する「文書×単語」のデータセットが大規模になると計算コストが高くなることが挙げられる。また，PLSAもLSAやNMFと同様に，結果が観測データに完全に依存するため，過学習を起こしやすく，新しい文書のトピックは推定できないという課題があるが，一方で，観測データに対する再現度は高く，その観測データ固有の特徴をそのまま反映できるモデルと捉えることもできる。

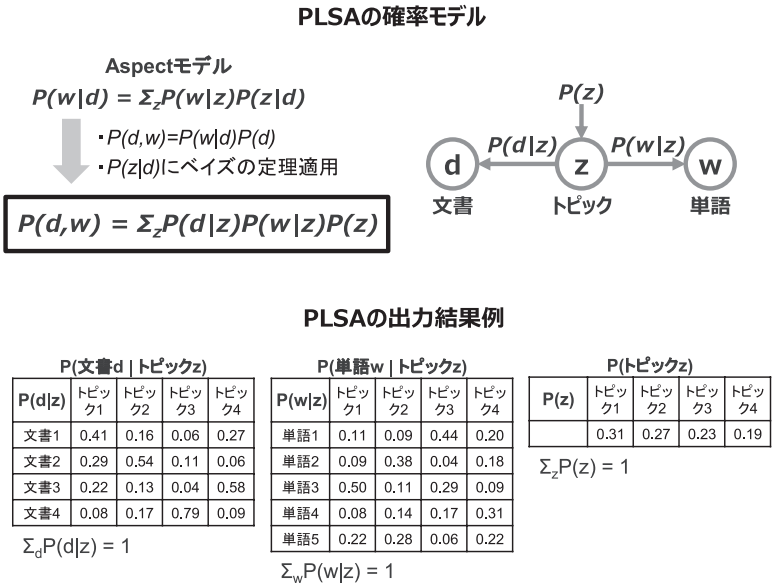


図7　PLSAの確率モデル

3.4　LDA

LDA（Latent Dirichlet Allocation，潜在ディリクレ配分法）は，PLSAをベイズ的に拡張させた手法で，ディレクレ分布を事前分布に導入しており，2003年に発表された[6)]。トピックモデルの中では最もよく使われる手法といえる。

LDAのグラフィカルモデルを図8に示す。LDAは確率的な生成モデルともいわれ，文書d内の各単語wが特定のトピックzから生成されたと仮定する。このトピックzの分布は文書dごとに異なり，その確率分布P（z|d）はθで表される。一方，特定のトピックzが各単語wを生成する確率分布P（w|z）はϕで表し，そのトピックzの下で単語wを生成するプロセスではϕが参照される。つまり，LDAでは$\theta \rightarrow z \rightarrow w$という生成プロセスが仮定され，その中で$z \rightarrow w$というアローの確率過程は$\phi \rightarrow w$というアローによる確率分布$\phi$が参照される。なお，$\theta$と$\phi$はそれぞれハイパーパラメータ$\alpha$と$\beta$を持つディレクレ分布に従う。LDAの推定対象は$\theta$＝P（z|d）と$\phi$＝P（w|z）であるが，これらの生成過程を逆にたどるアルゴリズムにより，単語w（観測データ）からθとϕを推定する。具体的には，各文書d中の各単語w（観測データ）に対して，まずその単語wがどのトピックzから生成されたと考えられるか，θとϕの暫定値（初期値）から推定し（zの推定），そしてその結果を基にして，各文書dがどのトピックzをどれだけ含むか推定し（θの推定），各トピックzがどの単語wを生成する可能性が高いか推定する（ϕの推定）。これを繰り返すことでθとϕを逐次的に更新していく。推定アルゴリズムは，ギブスサンプリングや変分ベイズ法などが適用され，反復的に計算される。

PLSAはパラメータが固定的で，観測データのみから直接推定するため，結果が完全に観測データに依存するが，LDAはパラメータが事前分布に従って変動する確率分布とし，観測データと事前分布から推定する。事前分布を導入することで，確率的なスムージング効果があり，過学習を抑制できる。また，LDAは新しい文書についても推定ができ，新しい文書に対応するθを未知とし，新しい文書に含まれる単語w（観測データ）と学習済みのϕとαから

θ を推定する。

一方，LDA はハイパーパラメータ α と β によって結果が変動しやすく，この値の設定の仕方，推定の仕方が難しい。ハイパーパラメータ α，β の大きさとディレクレ分布の特徴は，α，$\beta > 1$ では，値が大きいほど確率が均一化し，どの文書も同じようなトピックの分布を持つようになり，トピックは多様な単語に分散して関連づけられる傾向がある。α，$\beta = 1$ では分布は一様分布となり，様々なトピックにランダムに確率が割り当てられる。これは事前分布を仮定していない PLSA と近い振る舞いをする。$1 > \alpha$，$\beta > 0$ では，値が小さいほど確率が局所的になり，文書は特定のトピックに確率が集中し，トピックは一部の単語に偏って関連づけられる傾向がある。また，LDA は新しい文書の推定ができる高い汎化性能を持つ一方で，トピックの結果が一般的で抽象度が高くなることがある。

なお，PLSA と LDA はどちらも文書 d と単語 w の関係をモデル化している手法だが，文書 d と単語 w の関係の扱い方の違いには注意が必要である。PLSA の推定対象は P（d|z）と P（w|z）であるが，「文書 d ×単語 w」の行列データに対して文書 d と単語 w の役割には対称性があり，行と列を入れ替えても推定結果に影響はない。一方，LDA の推定対象は P（z|d）と P（w|z）であるが，文書 d と単語 w の役割は対称的ではなく，それぞれの役割は明確に区別されたモデルとなっている。そのため，行と列を入れ替えて適用すれば異なる結果となる。

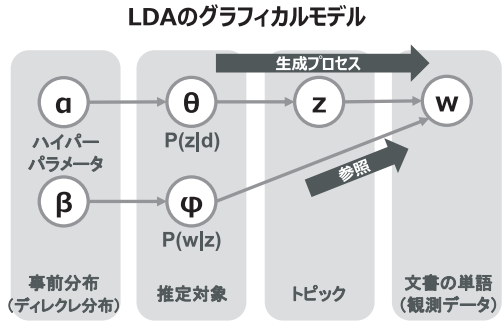


図 8　LDA のグラフィカルモデル

4.　深層学習モデル

続いて，第 3 次 AI ブームにおける「深層学習モデル」を応用した自然言語処理技術の各手法について解説する。第 3 次 AI ブームの火付け役となった深層学習モデルは，まずは画像認識の分野を中心に 2006 年頃から大きく発展し，特に 2012 年からは実用面での成果を次々に上げ，社会的な AI ブームとして認知されるようになった。一方，この頃より自然言語処理の分野でも深層学習モデルの応用が進められた。2013 年に発表された「word2vec」を皮切りに，深層学習を応用することで機械翻訳や文章分類，文章生成といった多くの自然言語処理タスクにおいて性能が向上した。特に 2017 年に Google とトロント大学の研究者によって発表された「Transformer」は大きなブレイクスルーとなり，そのアーキテクチャをベースに膨大なテキストデータを学習させた，汎用的な大規模言語モデル（Large Language Model, LLM）というものが生まれた。そして大規模言語モデルに基づいて文章を生成する「ChatGPT」という AI チャットボットが OpenAI によって 2022 年 11 月に発表された。その賢すぎる AI に世界は騒然とし，生成 AI ブームという新たな潮流が生まれ，巨大 IT 企業を中心に大規模言語モデルの開発競争が激化した。このように，深層学習によって始まった第 3 次 AI ブームは，Transformer の誕生によって新たなフェーズの幕開けとなった。

ここでは第 3 次 AI ブームの火付け役となった「深層学習」についておさらいをした上で，自然言語処理の分野における第 3 次 AI ブーム前半の代表的な技術として，「word2vec」「RNN」「LSTM」「seq2seq」「Attention」を，また第 3 次 AI ブーム後半の代表的な技術として「Transformer」「BERT」「GPT」「T5」を取り上げ，それぞれの概要を解説する。特に自然言語処理に大きな変革をもたらした Transformer とそれ以後の技術についてはより詳細に解説する。

4.1　深層学習

深層学習の概念は 1980 年代あるいはそれ以前から存在していたが，2006 年にトロント大学のジェフリー・ヒントン氏らの研究チームが重要な技術的進歩を発表したことで，これが第 3 次 AI ブームの引き金となった。ここでは第 3 次 AI ブームにいたる歴史を振り返りながら，その中心的な技術である深層学習の概要を解説する。

深層学習は多層のニューラルネットワークを利用して，大量のデータから学習し，高度なパターン認識や予測を行う手法であり，ディープラーニング（Deep Learning）とも呼ばれる。ニューラルネットワークは入力層，隠れ層（中間層），出力層で構成され，各層の処理では，前の層からの情報を線形変換と非線形変換をして次の層に引き継いでいる。線形変換とは前の層からの情報に重みをかけてバイアスを加えた加重和の計算であり，非線形変換とはシグモイド関数やハイパボリックタンジェント関数といった活性化関数を適用する処理であり，より複雑な表現を可能にしている。特に隠れ層を多層構造にしたモデルは深層ニューラルネットワーク（Deep Neural Network, DNN）と呼ばれ，多くの隠れ層を持つことでより複雑な表現能力を獲得でき，性能が向上すると考えられていた。そしてヒントン氏らは 1986 年に，ニューラルネットワークの予測誤差の情報を出力層から入力層へ逆方向に戻しながら各層の重みとバイアスを更新する誤差逆伝播法（Backpropagation）を発表し [7]，多層のネットワークでも効率的な学習が可能であることを示した。

誤差逆伝播法は，ニューラルネットワークで出力された予測結果と実際のデータとの誤差を最小にするように各層の重みを更新する方法であり，その誤差を表す損失関数を最小化するように各層の重みが勾配降下法により更新される。勾配降下法では，まずどの方向に各層の重みを調整すれば損失関数の値を効率的に減少できるのか知るために，各層の重みに対する損失関数の勾配（偏微分）を計算する。勾配の計算では，合成関数の微分を行う連鎖律（Chain Rule）を適用し，出力層から入力層に向かって各層の重みにおける損失関数の勾配（偏微分）を逐次的に計算する。その各層の重みにおける勾配を集合したベクトル（勾配ベクトル）は損失関数の正の変化率が最も大きな方向を示すため，それを負にした逆方向のベクトル，すなわち損失関数が最も降下する方向に向けて重みを更新していく方法が勾配降下法となる。なおこの更新は解析的には求められないので，反復計算を通じて近似的に損失関数が最小になる重みを見つけていく。

2006 年以前は深層ニューラルネットワークを実現する上で大きな問題が主に 2 つあった。一つは誤差逆伝播法における勾配消失と勾配爆発の問題である。誤差逆伝播法では，損失関数の勾配を求める際に，出力層から入力層に向かって各層における勾配を連鎖律で乗算して計算していく。勾配消失は，各層を通じて微小な勾配が乗算され続けることで，勾配が急速に小さくなり，入力層に向かって勾配がゼロに近づく現象である。これにより，入力層に近い層の重みが更新されないという問題が生じる。勾配爆発は，逆に大きな勾配が乗算され続けることで勾配が急速に大きくなる現象であり，重みの更新において安定性と収束性が損なわれる問題が生じる。もう一つの問題は過学習であり，隠れ層が増えることでモデルが複雑になり，特定のデータに過剰に適合し，新しいデータに対する汎化性能の低下が生じる。これらの問題は，深層ニューラルネットワークの効率的な学習を妨げる主な要因となっていた。

そして 2006 年にヒントン氏らは深層学習における 2 つの重要な論文を発表した。一つ目は深層ニューラルネットワークを高速に学習できるアルゴリズムに関するものであり [8]，もう一つはオートエンコーダを用いた次元削減と特徴抽出の方法である [9]。オートエンコーダは，入力層と出力層を一致させたニューラルネットワークで，データの圧縮（エンコーダ）と復元（デコーダ）を学習して隠れ層で次元を削減する方法である。これを事前学習することであらかじめデータの重要な特徴を抽出しておき，それを後続の学習タスクに活用することで学習の効率化と性能の向上に貢献した。さらに，このオートエンコーダは先述した 2006 年以前の 2 つの問題の緩和にも貢献した。ヒントン氏らによって提案されたオートエンコーダの事前学習は，一度にすべてを学習するのではなく，各層ごとに分割して学習し，それを次の層に連携していく。これにより誤差逆伝播法を実行する際は，勾配が全層を一度に通過するのではなく，段階的に伝播されるため，特に勾配消失の問題において影響を受けにくくなった。なお，勾配爆発の問題も緩和効果はあるが，誤差が大きい場合の学習では依然として大きな勾配となる可能性はある。またオートエンコーダは次元削減でデータの汎用的な特徴を抽出するため，過学習のリスクを低減させることができた。こうした深層学習の実現に関して重要な発表があった 2006 年以降，深層学習の研究は大きな発展を遂げた。

その後，2012 年 9 月に開催された画像認識の競技大会 ILSVRC（ImageNet Large Scale Visual Recognition Challenge）で，ヒントン氏率いる研究チームが開発した AlexNet[10] が圧倒的な性能を示した。また同 2012 年 6 月には Google がネット上の大量の画像データに深層学習を適用することで，コンピュータが猫を認識できるようになったと発表した。特にこれらの深層学習モデルでは，1998 年にヤン・ルカン氏らによって発表された畳み込みニューラルネットワーク（Convolutional Neural Network, CNN）という，画像の局所的なパターンを効率的に認識できる手法 [11] が採用された。このように 2012 年以降は，特に画像認識の分野を中心に，深層学習モデルが実用面において大きな成果を次々に上げていき，第 3 次 AI ブームが社会的にも認知されるようになった。

ここからはこの深層学習モデルを自然言語処理の分野で応用して開発された各手法について解説していく。

4.2　word2vec

word2vec は大量のテキストデータ（コーパス）を用いたニューラルネットワークモデルで，単語のベクトル表現を得る手法であり，2013 年に Google によって発表された [12]。このベクトルは分散表現，あるいは単語の埋め込みベクトル（Word Embedding Vector）と呼ばれ，単語の意味や類似性を捉えることができており，単語同士の演算もできる。

word2vec は隠れ層が 1 層となるシンプルな深層学習モデルを採用しており（そのため一般的な「深層」モデルとは異なるが），CBoW と Skip-Gram の 2 つのアルゴリズムがある。CBoW（Continuous Bag of Words）は，ある単語をその周辺単語から予測し，Skip-Gram はある単語からその周辺単語を予測する。それぞれのイメージを図 9（上）に示す。つまり，word2vec は深層学習により大量のコーパスの穴埋め問題をひたすら解いており，この事前学習されたモデルを使うことで，単語のベクトル表現を容易に得ることができるというものである。なお学習の中で用いられる周辺単語の数は「ウィンドウサイズ」と呼ばれるパラメータとして指定し，5 から 10 の範囲で設定することが一般的である。

word2vec の代表的なモデルには，たとえば Google News の約 1000 億語から成るニュース記事を元に学習された「GoogleNews-vectors-negative300」があり，約 300 万語に対して，入力した単語の 300 次元のベクトルが得られるモデルである。word2vec のイメージをするにあたって，仮に単純な 4 次元のベクトルで考えると，図 9（下）に示すような単語のベクトル表現が得られる。さらに，word2vec は線形なモデルであり，単語の意味的な演算をすることもできる。これは隠れ層が 1 層のシンプルなネットワーク構造から単純な線形変換を行い，単語は線形写像を通じて多次元ベクトル空間に配置されることで，得られる単語ベクトルが線形性を持つためである。たとえば王のベクトルから男のベクトルを引いて女のベクトルを足すと女王のベクトルになるといった演算が成り立つ。

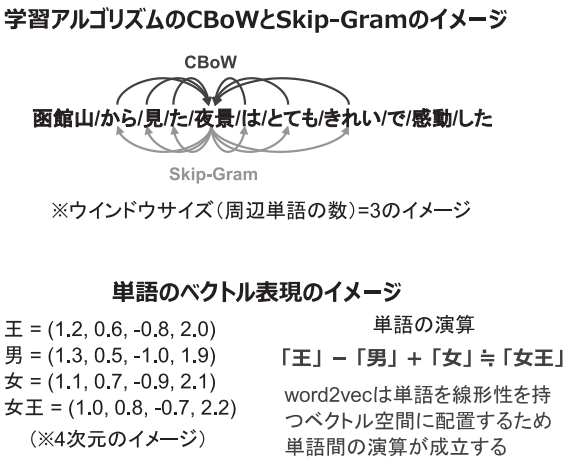


図 9　word2vec の学習と単語のベクトル表現

また，各文書に対して単語と同様に「文書 ID」の要素を持たせて word2vec の学習を適用することで，単語と同様に「文書 ID」のベクトル表現を得ることができる。この手法は doc2vec と呼ばれ [13]，文書全体のベクトルを生成することができる。

word2vec はこれまでの自然言語処理技術とは一線を画した革新的なもので，当時は一世を風靡した。ただ，word2vec は単語の順序情報が欠落していて，同じ単語でも文脈によって意味が異なる場合は区別できないという課題が指摘された。

4.3 RNN

RNN（Recurrent Neural Network，再帰型ニューラルネットワーク）は系列データを処理するモデルであるが，自然言語処理分野の適用では，文章を単語の系列データとして捉え，単語を一つずつ順番に逐次的に処理する。通常の深層学習はフィードフォワード型と呼ばれ，入力から出力に向けて一方向に流れるが，RNN では，隠れ層からの出力が再度自分の入力に帰る再帰型（リカレント）の構造をしていることが最大の特徴である。RNN の構造のイメージを図 10（上）に示す。

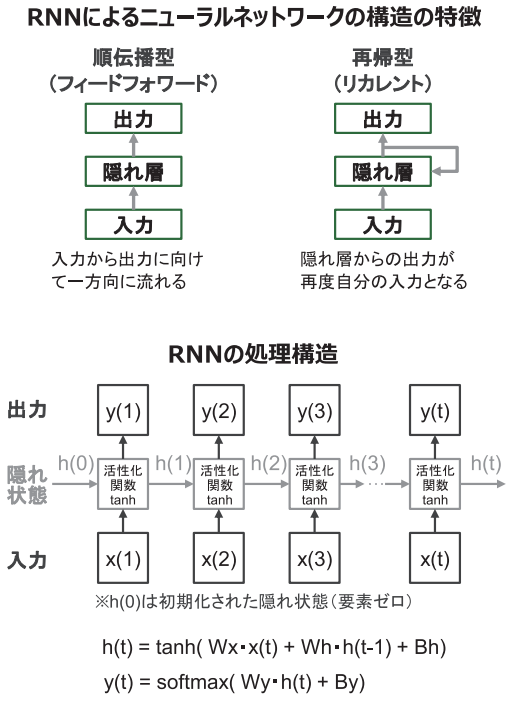


図 10　RNN の構造と処理の流れ

RNN は明確な起源を示すことが難しいが，その歴史をたどると，最初にその考え方に影響を与えたのは，ジョン・ホップフィールドによって 1982 年に発表された論文であると考えられる [14]。この論文ではフィードフォワード型のニューラルネットワークではなく，全結合型のニューラルネットワークによる処理方法が提案され，ホップフィールドネットワークと呼ばれる。その後シンプルな RNN の構造として，1986 年に発表された出力層の結果をフィードバックするジョーダンネットワークや [15]，1990 年に発表された隠れ層の出力をフィードバックするエルマンネットワーク [16] があり，これらは単純再帰型と呼ばれる。2010 年にはトマス・ミコロフによって自然言語処理の分野で RNN を適用し，次の単語を予測する言語モデルが発表され [17]，2010 年代初めから自然言語処理タスクにおいて RNN が盛んに適用されるようになった。

RNN の処理は，各ステップの隠れ層における隠れ状態の情報が次の隠れ層の入力にもなる再帰的な処理をしている。RNN の処理のイメージを図 10（下）に示す。より具体的な処理の流れは，t 時点のステップにおいて，入力ベクトルの x（t）に加え，前の隠れ層から h（t-1）という隠れ状態ベクトルの情報を受け取り，それぞれに重み行列

Wx と Wh を掛ける。それにバイアス項 Bh を加えたものに活性化関数 tanh を適用することで隠れ状態ベクトル h(t) を出力する。これは次の隠れ層に送る情報となる。さらにこの隠れ状態ベクトル h（t）に重み Wy を掛けたものにバイアス項 By を加え，これに活性化関数 softmax を適用することで出力ベクトル y(t) を出力する。なお，それぞれのバイアス項は出力を調整するパラメータである。活性化関数 tanh はハイパボリックタンジェント（Hyperbolic tangent）という関数で，これは入力値を－ 1 から 1 の間に変換する関数である。類似する関数に入力値を 0 から 1 に変換するシグモイド関数があるが，tanh 関数はシグモイド関数よりも勾配が大きく，誤差逆伝播法における勾配消失問題が起きにくいとされる。また活性化関数 softmax はベクトルの各成分の合計が 1 となるように変換する関数で，各成分の値は 0 から 1 までの値を取るため，その出力は確率分布とみなすことができる。出力ベクトル y（t）は，学習データから得られた出力単語の候補群（語彙）における確率分布となっており，その確率の最も高い単語が最終的に選択されることになる。そしてその出力結果と正解の誤差情報から誤差逆伝播法によって各重みとバイアスのパラメータが更新される。なおこれらのパラメータは各系列ステップ間で共通している。

word2vec では単語の順序情報を考慮することができなかったが，RNN では過去の処理情報を保持して，それが現在の入力に影響を与える構造により，単語の順序（文脈）を考慮した処理ができることが特長となる。

しかし，長い系列データの場合，過去の情報を反映させるのが困難となる「長期依存性の問題」が指摘されている [18]。これは，再帰的処理により隠れ状態が次の隠れ状態に連携されるが，長い系列でネットワークが深くなるとこの処理が繰り返され，誤差逆伝播法において勾配爆発や勾配消失が起き，過去の遠い情報を最適に保持できなくなるという問題である。勾配爆発では，再帰処理により，同じ重みが繰り返し乗算され，勾配が指数関数的に増加し，勾配消失では，再帰処理により，微分が 0 に近い活性化関数を繰り返し通過し，勾配が急速に小さくなる。

4.4 LSTM

LSTM（Long Short-Term Memory，長・短記憶）は RNN と同じく系列データの処理モデルであるが，RNN で課題となっていた長い系列の処理を可能とするモデルで，1997 年に発表された [19]。RNN と同様に LSTM も自然言語処理の分野において盛んに適用されるようになったのは 2010 年代初めの頃からであり，これは近年のビッグデータの利用やコンピュータの計算能力の向上が寄与している。

LSTM の特徴は，RNN の隠れ状態 h（短期記憶）に加え，メモリセル c（長期記憶）の情報がセルからセルに受け継がれる仕組みを取っていることにある。このメモリセル c は記憶を長期に保持する機能を持っている。LSTM の処理構造のイメージを図 11（上）に示す。さらに LSTM の内部では，忘却ゲート，入力ゲート，出力ゲートという 3 つのゲート構造を持ち，各ゲートが情報の流れを制御している [20]。LSTM で情報の伝達を制御する内部構造のイメージを図 11(下)に示す。ここではそれぞれのゲートの特徴を述べるに留め，具体的な計算式の解説は割愛する。忘却ゲートは，受け継がれたメモリセル c(t-1）の長期記憶の情報のうちどれを受け継ぎどれを消去するか制御する。入力ゲートは，現在の入力 x（t）と前の隠れ状態 h（t-1）を使って，新しく記憶するべき情報の候補を生成し，どの情報を入力し記憶させるか制御する。出力ゲートは，次の隠れ状態 h（t）と出力 y（t）の情報を制御する。忘却ゲート，入力ゲート，出力ゲートでは，それぞれ最初に入力 x（t）と前の隠れ状態 h（t-1）に活性化関数のシグモイド関数を適用している。シグモイド関数の出力は 0 から 1 の値を取るため，どの情報をどの程度通過させるか，あるいは通過させないかという，オン・オフの制御として機能している。入力ゲートはこれに加えて tanh 関数の適用があり，新しく記憶するべき情報の候補を生成している。この tanh 関数の処理は RNN で次の隠れ状態を出力するプロセスと同様である。また，tanh 関数は－ 1 から 1 までの値を取るため，符号を司る機能があり，その情報が正の影響を与えるか負の影響を与えるかも決定する。入力ゲートではこれにシグモイド関数の適用結果と掛け合わせる処理を行うことで，どの新しい情報をどの程度記憶させるか制御している。そして入力ゲートでは，これと忘却ゲートで通過した情報を加えて次のメモリセルへと更新している。最後に出力ゲートでは，通過したメモリセルの情報に tanh 関数を適用し，それをシグモイド関数の適用結果と掛け合わせることで，長期記憶のうちどの情報を次の隠れ状態 h（t）と出力 y(t）に反映させるのか制御している。

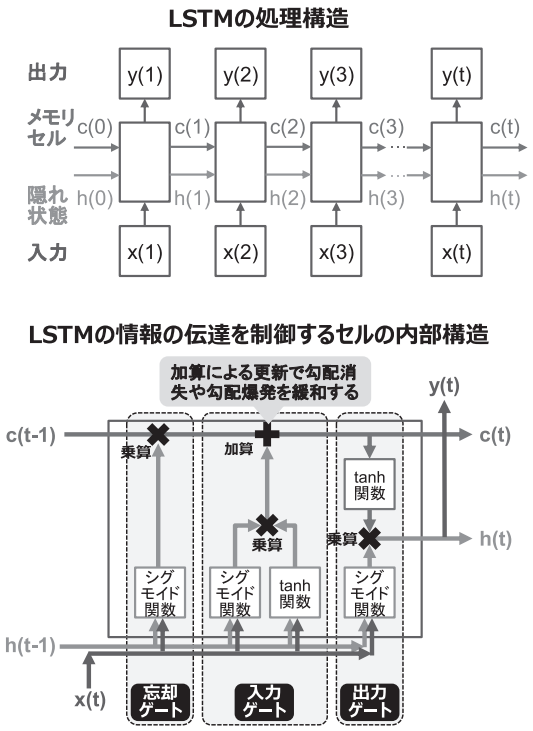


図 11　LSTM の処理とモデルの構造

RNN では隠れ状態の更新が重み行列の乗算と活性化関数の適用のみで行われ，これが長い系列で繰り返されることで長期依存性の問題が生じていた。一方，LSTM では，3 つのゲートで RNN と同様に重みの乗算と活性化関数が適用されるが，特に入力ゲートを通過する際には，忘却ゲートの結果と入力ゲートの結果を加算することでメモリセルが更新される。この加算による更新メカニズムにより，必要な情報を長期間セル状態に保持し続けることができ，長期依存性の問題を（特に勾配消失において）緩和することができた。

このように，LSTM は RNN よりも長い系列が扱えるとされるが，実際は完全ではなく，RNN ではうまく処理できる単語数が 10 語程度であるのに対し，LSTM でも約 20 語以上となると翻訳精度が劣化するといわれている [21]。また，LSTM は計算コストが高いという課題もある。

4.5　seq2seq

seq2seq は，系列データを系列データに変換するモデルであり，2014 年に発表された [22]。自然言語処理においては，seq2seq の代表例として NMT（Neural Machine Translation）があり，NMT は機械翻訳を主な用途としたもので，これも 2014 年に発表された [23]。seq2seq はエンコーダ-デコーダモデル（Encoder-Decoder model）とも呼ばれ，文章をベクトルに変換するエンコーダ（seq2vec）と，ベクトルを文章に変換するデコーダ（vec2seq）が連結した構造を持ち，これにより機械翻訳などの文章から文章への変換を実現している。

RNN や LSTM はそれ単体では，① seq2vec，② vec2seq，③入力・出力の系列数が一致する seq2seq（たとえば単語の品詞割当など）は処理できるが，機械翻訳のように系列数の一致しない seq2seq は処理できない。そこでこうした seq2seq を処理するために，RNN や LSTM による seq2vec（エンコーダ）と vec2seq（デコーダ）を連結して対応させる。seq2vec は，文章を一つのベクトルに変換する処理であり，RNN や LSTM の最後の隠れ状態ベクトルに該当する。一方，veq2seq は，一つのベクトルを文章に変換する処理であるが，RNN や LSTM では，一つのベクトルを入力として単語を出力し，その出力された単語と前の隠れ状態ベクトルを新たな入力として次の単語を出力することで，最終的に文章を生成する。RNN を用いた場合の seq2seq の処理構造を図 12 に示す。なお，RNN や LSTM が seq2seq のうち，単語の品詞割当など入力と出力の系列数が一致すれば処理できるが，機械翻訳など入力と

出力の系列数が一致しないと処理できない理由は，RNN や LSTM は系列ステップごとに出力を生成する性質があるためである。

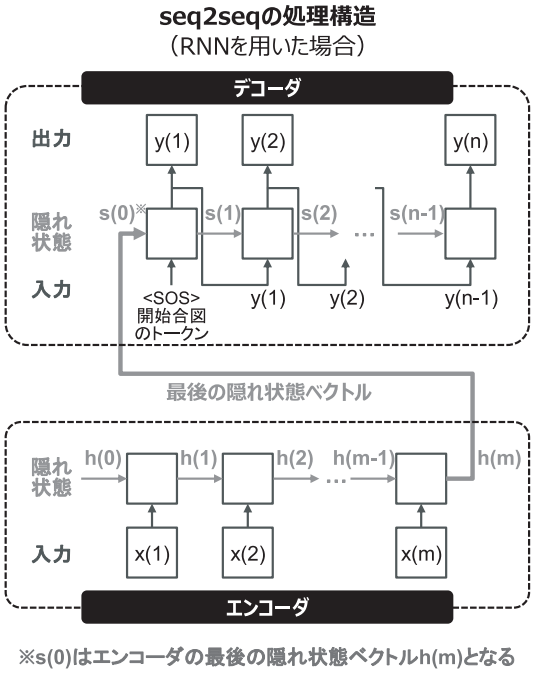


図 12　seq2seq の処理構造

4.6　Attention

Attention（注意機構）は，seq2seq において入力文章を出力文章に変換する際に，文章の中でどの単語に注目すべきかを判断する仕組みであり，2014 年に NMT の中で発表された [23]。

従来の seq2seq は，エンコーダの最後の隠れ状態である一つの固定長ベクトルだけがデコーダに連携されるため，受け渡しできる情報量に限界があり，長い系列ほど精度が落ちる課題があった。これに対し Attention では，入力文章を出力文章に変換する際に，入力のどの単語に注目すべきか，あるいは無視すべきかを考慮できるため，入力と出力の依存関係を捉えることができ，精度の高い出力を実現することができた。遠い所にある単語も含め，入力文章に含まれるすべての単語を参照できるため，RNN や LSTM で課題となっていた長期依存性の問題も補完できる仕組みとなっている。

Attention の特徴とモデル構造のイメージを図 13 に示す。Attention の仕組みを用いたモデルでは，デコーダの今の隠れ状態に対するエンコーダの各隠れ状態の注目度（Attention weight）を計算し，その重み付きの隠れ状態を足し合わせることで文脈ベクトルを作成し，これをデコーダの次の隠れ状態の入力として受け渡す。この処理をデコーダが新しい単語を出力する度に行い，新たな文脈ベクトルが作成され受け渡される。なお，Attention weight はデコーダとエンコーダの隠れ状態のペアを入力とする単純なニューラルネットで計算される。このニューラルネットでは，デコーダの隠れ状態とエンコーダの隠れ状態を入力としたときのそのペアの類似度 e がスカラー値として出力される。そこで得られた各ペアの類似度 e に softmax 関数を適用して合計が 1 となる重み a を計算すればそれが Attention weight となる。このニューラルネットにおいて，デコーダの隠れ状態の重み行列，エンコーダの隠れ状態の重み行列，隠れ層から類似度 e を出力する重み行列は，入力のソース文章と出力のターゲット文章のペアを教師とした学習過程を通じて，損失を最小化させる誤差逆伝播法によって最適化される。

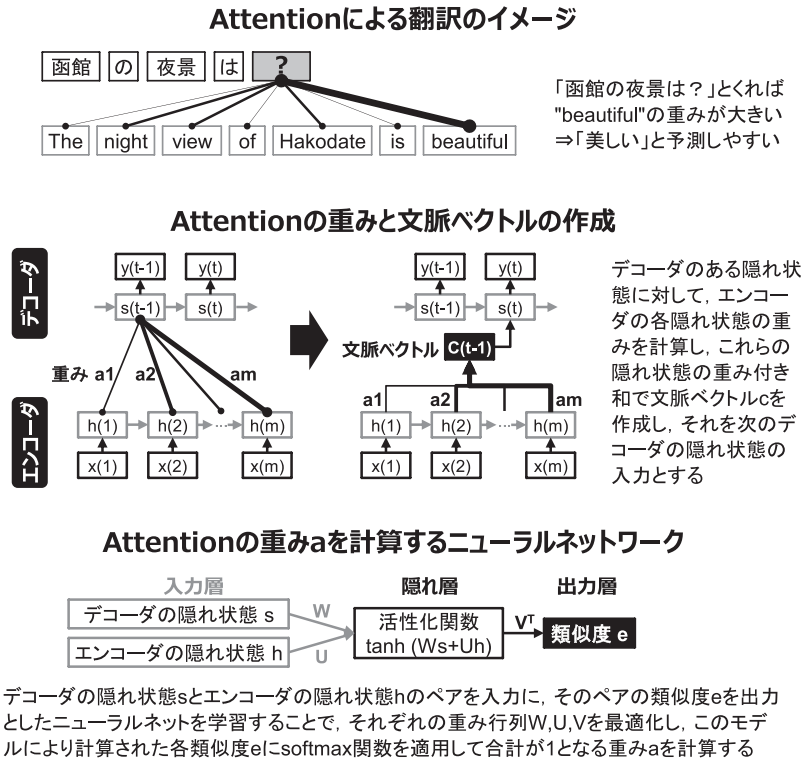


図 13　Attention の特徴とモデル構造

4.7　Transformer

Transformer は，2017 年に NeurIPS（Neural Information Processing Systems）という AI 分野のトップカンファレンスで，"Attention is All You Need" というタイトルで Google とトロント大学の研究者によって発表された手法である [24]。タイトルのとおり Attention だけを用いたエンコーダ - デコーダ構造の深層学習アーキテクチャであり，従来主流であった RNN や CNN の仕組みを使わないアプローチとして提案された。従来の seq2seq は RNN や LSTM を連結した構造となり，一つひとつ単語を入力して処理をするという逐次的処理で並列化計算が難しかったが，Transformer は一度に複数の単語を入力して並列化処理が可能になり，学習にかかる時間を大幅に短縮させることができた。高精度でかつ計算効率が非常に高い Transformer は，第 3 次 AI ブームの自然言語処理技術において大きなブレイクスルーとなった。非常に大規模なデータの学習も可能となったため，後に提案される BERT，GPT，T5 といった汎用大規模言語モデルは，その中核技術として Transformer が用いられた。

Transformer が高い精度と高い計算効率を実現している原理を理解する上で，そのアーキテクチャの構造において，特に 3 種類の Attention の仕組みが中心的な役割を果たしているとされている [21]。これらの処理により表現力の高いベクトルを獲得しながら，行列内積計算によって高速計算が可能になった。以下にその 3 種類の Attention の仕組みの概要を解説する。

一つ目は Self-Attention という仕組みである。元々の Attention は，入力文章（Source）の単語と出力文章（Target）の単語を対応づけた Source-Target 型の Attention であったが，Self-Attention は，入力文章と出力文章のペアではなく，同一文章の中の各単語が他の単語とどの程度関係しているのかを評価する。これにより，単語間の依存関係を学習することができ，より表現力の高いベクトルを獲得することができた。

二つ目は Scaled Dot-Product Attention という仕組みである。これは Self-Attention において，単語間の内積の類似度計算により他の単語との関連性を考慮した単語ベクトルを獲得する仕組みである。具体的な処理としては，各単語を固有の埋め込みベクトルに変換し，この単語ベクトルに対して，三つの異なる重み行列を掛けて線形変換を施した，「クエリ」「キー」「バリュー」という三つのベクトルを作成する。各単語のクエリに対して文中の他のすべての単語

のキーとの類似度を，それぞれのベクトルの内積によって計算する。このとき，ベクトルの次元数が大きいと内積の値が大きくなりすぎるため，実際はその内積を次元数の平方根で割ってスケーリングする。そしてすべての単語に対して計算したスケーリング済みの内積の合計が 1 となるように softmax 関数を適用し，これが Attention の重みとなる。この重みを各単語のバリューに掛け，それらすべてのバリューの重み付き和を取ったベクトルを元々の単語の新たな表現ベクトルとして生成する。つまり，それぞれの単語は文中の他の単語との類似度を反映したベクトルに変換されるということである。なお，単語の埋め込みベクトルや各重み行列などのパラメータは，モデルの学習過程で誤差逆伝播法によって，予測単語と正解単語の差異を最小にするように最適化される。Scaled Dot-Product Attention の処理構造のイメージを図 14（上）に示す。

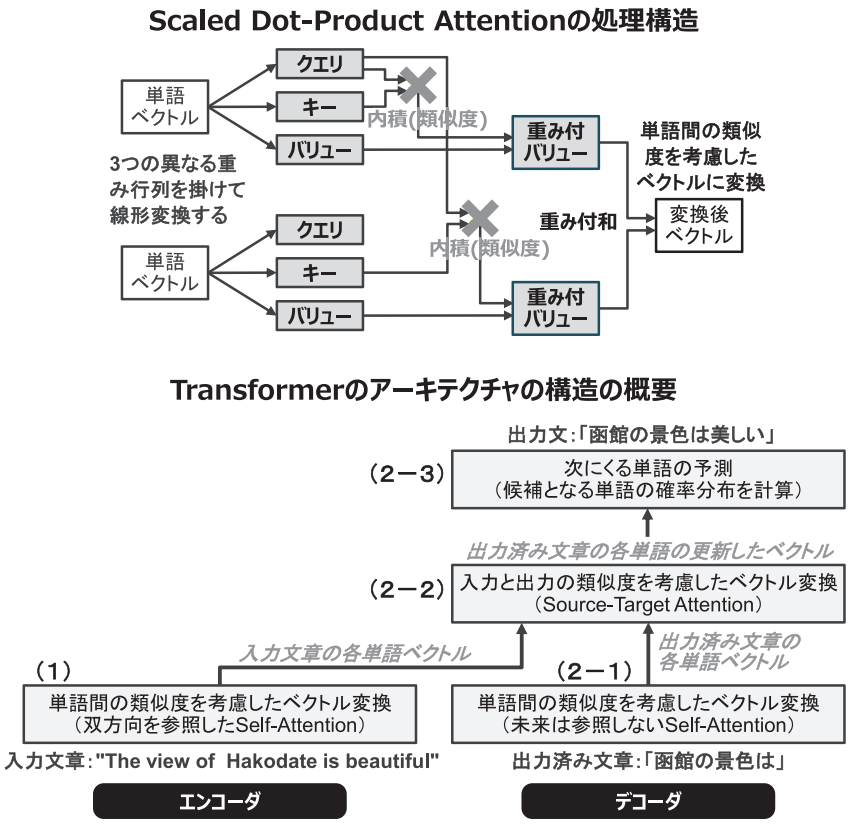


図 14　Transformer の Scaled Dot-Product Attention とアーキテクチャ構造

三つ目の重要な Attention の仕組みは Multi-Head Attention というもので，これは Self-Attention を異なる表現空間（ヘッド）で複数パターン実行し，より柔軟な単語ベクトルを獲得するというものである。先述した Scaled Dot-Product Attention の説明では，1 つの単語ベクトルに対して 1 組のクエリ，キー，バリューのベクトルを作成していたが，Multi-Head Attention では，1 つの単語ベクトルに対して複数のクエリ，キー，バリューのベクトルを作成する。それによって得られた複数の出力ベクトルを統合して，最終的に 1 つのベクトルに落とし込む。これにより複数の異なる観点からの類似度が考慮され，モデルの表現力を大きく向上させることができた。

Transformer のアーキテクチャを簡略化すると，(1) エンコーダと (2) デコーダの 2 つの処理プロセスに分けられ，デコーダのプロセスはさらに 3 つのステップに分けると分かりやすい。それは，(2-1) 出力済みの文章の単語をベクトル化するステップ，(2-2) 入力文章と出力済み文章の類似度に基づいて出力済み文章の単語のベクトルを更新するステップ，(2-3) 次にくる単語を出力するステップである。特に Transformer の特徴を示すステップが (1) (2-1) (2-2) である。図 14（下）に Transformer のアーキテクチャの構造の概要と処理プロセスのイメージを示す。なお，ここでは Transformer の構造を分かりやすく理解するための解説に留め，詳細な解説は Transformer の論文や他の専門書に委ねるとする。

まず（1）エンコーダの処理プロセスでは，先述の 3 種の Attention により，入力文章に存在するそれぞれの単語のベクトルをより表現力の高いベクトルに変換する。次はデコーダの処理プロセスに移る。まず，（2-1）出力済みの文章のそれぞれの単語をベクトル化するステップでは，エンコーダの処理と同様に，途中まで出力された文章において 3 種の Attention が適用され，出力済みの文章に存在するそれぞれの単語においてより表現力の高いベクトルが得られる。ただし，エンコーダの Self-Attention と違い，デコーダではまだ出力されていない未来の単語は参照できないというルールがあり，双方向に参照ができるエンコーダに対して，デコーダでは自身の単語よりも左側ある単語しか参照できない。次に（2-2）入力文章と出力済みの文章の類似度に基づいて出力済み文章の単語のベクトルを更新するステップでは，入力文章の各単語と出力済みの文章の各単語の類似度を考慮した Attention が適用される。この Attention は従来の seq2seq で適用されたソース・ターゲット型の Attention のアプローチが採用され，エンコーダ - デコーダ Attention とも呼ばれるが，ここでもクエリ，キー，バリューの 3 つのベクトルが用いられる。具体的には，デコーダで出力済みの文章（ターゲット文章）において出力された各単語ベクトルをクエリとし，エンコーダの入力文章（ソース文章）において出力された各単語ベクトルをキーとバリューとし，この 3 つのベクトルを使って Scaled Dot-Product Attention の処理を行う。これによりデコーダの出力済み文章の各単語に対して，入力と出力の類似度に基づいた新たな表現ベクトルが得られる。最後に（2-3）次にくる単語を出力するステップでは，（2-2）で更新されたベクトルに基づいて，次にくる単語が予測される。つまり，単語の予測には出力済みのすべての単語の情報に加え，入力文との依存関係の情報が利用されている。具体的な予測の処理は，デコーダで更新された出力済み単語の新たな表現ベクトルに対して，候補となる単語群（語彙）の重み行列を掛けた線形変換を施すことで，その候補単語のスコアが計算される。そして，それを softmax 関数で確率分布に変換し，最も確率の高い単語が選ばれる。そのため重要となる処理は予測したい単語の一つ前の単語，つまり出力済みの単語のうち右端にある単語のベクトルに対する予測処理である。なお，デコーダでまだ何も出力されていない最初の状態では，これから出力を開始する合図となる特別な開始トークン（たとえば< SOS >や< start >など）がデコーダに入力され，その開始トークンがエンコーダからの出力（各単語の文脈ベクトル）を受け継ぎ，次の単語（実質的には最初の単語）を予測することになる。Transformer のモデルは，この予測単語と正解単語の一致度を損失関数とした誤差逆伝播法により，モデル全体の各パラメータ（単語の埋め込みベクトルや各重み行列など）が最適に更新される。

また，Transformer は従来の RNN 構造を持たないため，Attention だけでは単語の順序情報を学習することができない。そこで Transformer は Positional Encoding と呼ばれる手法で，単語の埋め込みベクトルに順序情報を加えている。具体的には周期性のある sin 関数と cos 関数を使って何番目にある単語なのかという情報を含む位置情報ベクトルが生成され，それが最初の段階で各単語の埋込みベクトルに加算される。順序情報を単純な 1,2,3,4 といった連番ではなく，sin 関数と cos 関数を使用するメリットは，周期性がある関数によりシーケンスの長さに影響を受けないで順序情報を与えることができることが挙げられる。また，各単語間の相対的な距離や位置がすべて同じではなくそれぞれ区別された異なる数値によって表現されるため，単語間の複雑で微妙な位置関係もより豊かに捉えることができるというメリットも挙げられる。なお，この位置情報ベクトルは学習過程で最適化されるパラメータの対象ではない。

Transformer はもともと機械翻訳タスク向けに開発されたモデルだが，多くの自然言語処理タスクで応用が可能であり，どのケースでも非常に高い性能を実現した。また，自然言語処理の分野を超えて，Transformer は画像認識や画像生成の分野でも応用されている [20]。たとえば 2020 年に Google によって発表され，画像認識に Transformer を用いた ViT（Vision Transformer）や [25]，2021 年に OpenAI によって発表され，画像生成に Transformer を用いた DALL-E がある [26]。このように Transformer は多様な分野に応用されるほど，その情報処理能力の高さは驚異的であった。

4.8 BERT

BERT（Bidirectional Encoder Representations from Transformers）は，Transformer を使って大量のテキストデータを事前学習した汎用的な大規模言語モデルである。2018 年 10 月に Google によってプレプリントという形で発表

され[27]，その後 2019 年 6 月に開催された NAACL（North American Chapter of the Association for Computational Linguistics）という自然言語処理分野のトップカンファレンスで発表された[28]。2019 年 10 月には Google が検索エンジンに BERT を採用し，「過去 5 年間で最大の飛躍」と発表され話題となった。今まで検索キーワードに対して検索結果を返していたところを，文章の検索に対して結果を返せる精度を大幅に向上させた。

BERT は，Transformer のエンコーダ部分を使用したモデルであり，BERT の頭文字である Bidirectional（双方向）の意味は，Self-Attention を実行するときに，各単語は同一文中の左右にある他のすべての単語との関係性を捉えるということである。BERT の Self-Attention のイメージを図 15（上）に示す。ただ，これは通常の Self-Attention の仕組みである。BERT がわざわざ双方向を強調したのは，BERT より先の 2018 年 6 月に OpenAI によって発表された GPT と明確に区別する意図があるともいわれている[21]。GPT については次に解説するが，GPT は Transformer のデコーダ部分を使用したモデルであり，左から右の一方向の Self-Attention が実行され，文章生成などのタスクを得意としている。これに対して BERT はその双方向の Self-Attention により，文章全体の文脈理解において優れており，文章分類や感情分析などのタスクに適しているといわれる。

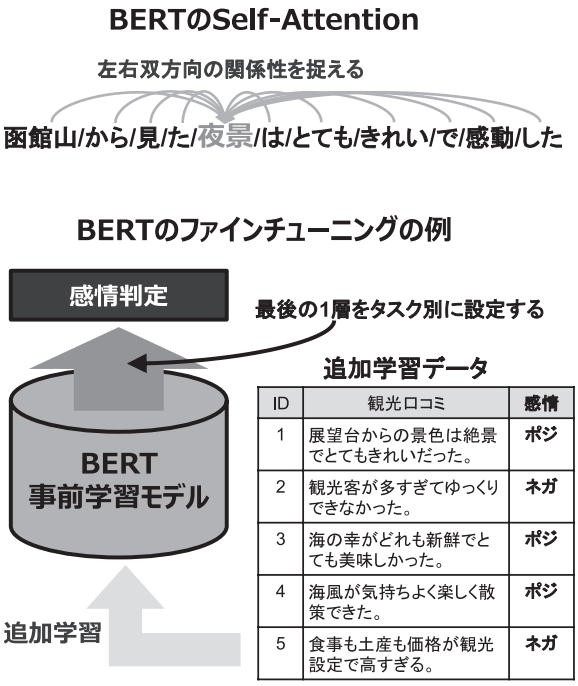


図 15　BERT の Self-Attention とファインチューニング

BERT の事前学習では，自己教師あり学習の MLM（Masked Language Model）と NSP（Next Sentence Prediction）が実行される。MLM は，文中の一部の単語をマスクし，その単語を双方向から予測するタスクである。この穴埋め問題の予測は Self-Attention によって得られる各単語の表現ベクトルを入力とした深層学習モデルが適用され，誤差逆伝播法によって Self-Attention における各重み行列が最適に更新される。これにより各単語が文中でどのように使われているのかという理解を深め，文脈を捉えた単語の表現を学習できる。MLM が単語単位の学習であるのに対し，NSP は，文単位の学習となる。NSP では 2 つの文が与えられたときに，それらが連続する文か否かを予測する。これにより文の流れや論理的なつながりを理解し，文章全体の構造を把握する能力を向上させる。BERT はこうした事前学習を通じて広範な文脈理解能力を獲得している。なお，BERT では，入力文章の先頭には [CLS] という特殊なトークンが挿入され，この CLS トークンも他の単語と同様に Self-Attention でベクトル表現が学習されるが，CLS トークンは文章全体の意味を表すような表現となる。NSP の事前学習では，2 文を連結した先頭に挿入された CLS トークンに対応するベクトルを用いて，それらが連続するか否かを判定している。

このように BERT では，文章全体の文脈を捉えた単語のベクトル表現を，あるいは CLS トークンに対応する文

章のベクトル表現を出力することができる。たとえば従来の word2vec も単語のベクトル表現を出力できるが，word2vec は同じ単語は同じベクトルが付与される。「マウスをクリックする」と「マウスで実験する」で登場する「マウス」は word2vec では同じベクトル表現となるが，BERT では文脈を考慮した異なるベクトル表現として得られる。なお，BERT には代表的なモデルに Base モデルと Large モデルがある。Base モデルは Transformer のエンコーダ層を 12 層重ねており，パラメータ数は 1.1 億個で 768 次元のベクトルを出力する。Large モデルは Transformer のエンコーダ層を 24 層重ねており，パラメータ数は 3.4 億個で 1024 次元のベクトルを出力する。

また，BERT の事前学習モデルを再利用し，個別のタスクに応じた比較的小規模な教師データで追加学習をするファインチューニング（Fine-tuning）を施すことで，その個別タスクに応じてモデルが微調整される。これによって，少量の学習データでも高い精度を得ることができる。その仕組みは，文章分類や感情分析，質問応答といった個別のタスクに対して，BERT の最後の 1 層だけそのタスクに特化したニューラルネットワークを追加する。つまり BERT で最終出力される単語や文章のベクトル表現を入力層とし，タスクの解答を示す出力層を追加したニューラルネットワークを BERT の最終層に構成する。そしてそのタスクの教師データで BERT モデル全体を再学習させれば，そのタスクに最適なモデルを構築できるというものである。BERT のファインチューニングのイメージを図 15（下）に示す。従来のように個別のタスクごとにモデルを構築しなくても，この事前学習モデル＋ファインチューニングというアプローチにより，個別のタスクを低コストでかつ高い性能で実現できるということが BERT の大きな成果といえる。BERT が発表されて以降，自然言語処理領域ではとりあえず BERT を試してみるという取り組みが多発した。

4.9　GPT

GPT（Generative Pre-trained Transformer）も Transformer を使って大量のテキストデータを事前学習した汎用的な大規模言語モデルであり，2018 年 6 月に OpenAI によって発表された [29]。GPT は Transformer のデコーダ部分を使用したモデルで，テキストの生成能力において優れており，文章生成や文章補完，要約作成，対話生成，言語翻訳などのタスクに適しているといわれる。GPT が Self-Attention を実行するときは，左側にある単語のみが使われ，一方向の関係性を捉える。事前学習は自己教師あり学習となり，テキストの次にくる単語を予測し，これによりテキストの生成能力を獲得している。GPT の Self-Attention のイメージを図 16（上）に示す。

GPT の事前学習モデルを再利用する際は，特定のタスクの説明や指示を入力（プロンプト）として与える。特に GPT-3 からは Few-Shot Learning という概念が注目された。これは入力に「ショット（shot）」と呼ばれるタスクを解く例（サンプル）を少数与えることで，ファインチューニングをすることなくさまざまなタスクに対応できるという，柔軟で効果的な学習能力である。与える例の数が少数（2 個以上）ある場合を Few-Shot，1 個ある場合を One-Shot，タスクを解く例を一つも与えない場合を Zero-Shot と呼ぶ。GPT の Few-Shot Learning の例を図 16（下）に示す。

なお，BERT のファインチューニングでは，特定のタスク用にモデル全体を再学習する手法であり，新しいタスク用の追加データセットを使用して，モデルの全パラメータを更新する。一方，GPT の Few-Shot Learning では，新しいタスクを解決するための直接的なヒントやガイダンスを提供しているだけで，モデルのパラメータは固定され，更新されない。GPT が Few-Shot Learning により新しいタスクに柔軟に適応できる理由は，その事前学習モデルが大規模なデータと大規模なモデル構造により学習されているためである。これにより言語パターンの高い汎用性と一般化能力を獲得し，少数の例からでもそのタスクの性質を理解し解決することを可能としている。ただし，さまざまなタスクに対応できる汎用性の高さがある反面，特定のタスクを対象に学習されたモデルと比べると性能が劣ることがある。また，与えられるプロンプトの依存性が高く，これがモデルの性能に大きく影響を与えるため，ショットの内容を含む適切なプロンプトの設計が求められる。このようにモデルの性能を上げるような良いプロンプトを設計することをプロンプトエンジニアリングと呼び，さまざまなコツや例文が提案されている。

OpenAI は 2019 年に GPT-2 を [30]，2020 年に GPT-3 を発表し [31]，2022 年 11 月には GPT-3.5 をベースとしたチャットボット“ChatGPT”をリリースした。ChatGPT の利用者数はリリース後 2 ヶ月で 1 億人に達し，その性能の高さに世界中が騒然とした。2023 年 3 月にはさらに進化した GPT-4 を公開し，2023 年 9 月には音声や画像のデータもテキストのデータと同時に扱えるマルチモーダルなタスクに対応可能にした。ChatGPT の発表を皮切りに生成

AI ブームが勃発し，巨大 IT 企業を中心に生成 AI の研究開発と新たなサービスの競争が激化した。

なお，初代 GPT は Transformer のデコーダ層を 12 層重ねたモデルで，Book Corpus と呼ばれる未発表書籍 7000 冊分以上のテキストデータ 4.5GB を学習し，モデルのパラメータ数は 1.2 億となっている。GPT-2 は Transformer のデコーダ層を 48 層重ねたモデルで，40GB の Web テキストのデータを学習し，モデルのパラメータ数は 15 億となっている。GPT-3 は Transformer のデコーダ層を 96 層重ねたモデルで，570GB の Web テキストのデータを学習し，パラメータ数は 1750 億個となっている。GPT3.5 以降は，筆者がこの原稿を執筆した 2024 年 3 月時点では，モデルの内容の情報は公式に発表されていないが，より大規模なデータでより大規模なパラメータ数の学習モデルが開発され続けている。

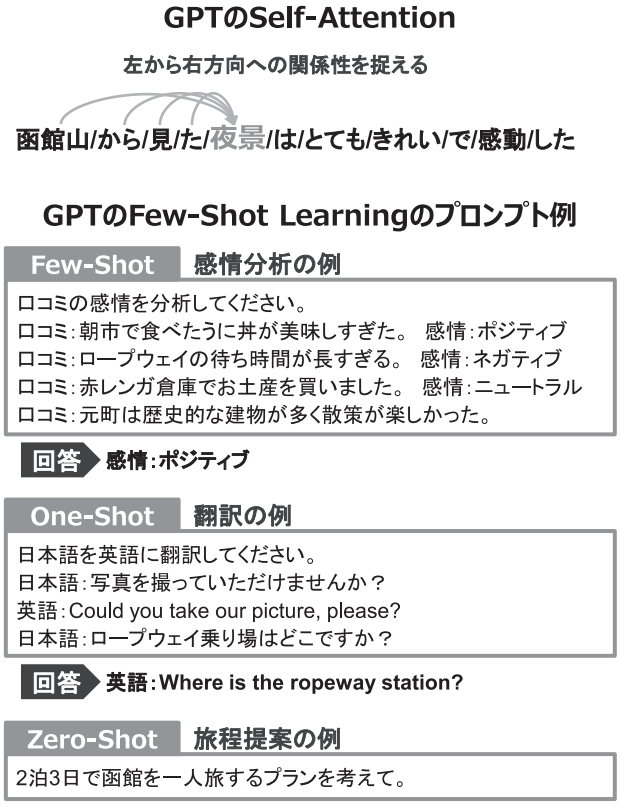


図 16　GPT の Self-Attention と Few-Shot Learning

4.10　T5

T5（Text-to-Text Transfer Transformer）も Transformer を使って大量のテキストデータを事前学習した汎用的な大規模言語モデルであり，2019 年 10 月に Google によって発表された [32]。T5 は Transformer のエンコーダ - デコーダ構造を持つモデルで，従来のように個別タスクごとにモデルを構築するのではなく，"Text-to-Text" とあるように，あらゆる自然言語処理タスクをテキストからテキストに変換する同一のアーキテクチャで扱うことができる。たとえば，「翻訳：英語の文→日本語の文」，「要約：長い記事→短い要約」，「感情分析：口コミ→感情判定」など，タスクの詳細を入力テキスト自体に組み込むことで，あらゆるタスクを「入力テキスト→出力テキスト」という形式に統一する。これは BERT のように，事前学習＋ファインチューニングというアプローチに基づき実現している。

T5 の事前学習も自己教師あり学習となるが，エンコーダ - デコーダ構造の学習であることがポイントである。BERT と GPT の自己教師あり学習と比較すると，BERT はエンコーダモデルであり，テキストの一部をマスクし，それを双方向の Self-Attention で予測することで，高い文脈理解能力を獲得した。GPT はデコーダモデルであり，左から右の一方向の Self-Attention によりテキストの次に来る単語を予測することで，高いテキスト生成能力を獲得した。

これに対して T5 では，まずエンコーダにおいて，元のテキストの一部を特殊なトークンに置換する。より正確には，スパン（span）と呼ばれる単語あるいは連続する単語のフレーズをトークンで置換し，一部が欠落したテキストを用意する。そのテキストを対象に，エンコーダでは双方向の Self-Attention によってそのトークンも含めて各単語のベクトルを出力する。デコーダでは，すでに生成済みのテキストに対して，左から右への一方向の Self-Attention を適用し，生成済みのテキストに含まれる各単語のベクトルを出力する。さらに，エンコーダから受け取った各単語のベクトルに対してソース・ターゲット型の attention を適用する。これによりデコーダで生成済みのテキストに含まれる各単語は，エンコーダとデコーダの類似度に基づき新たな表現ベクトルを獲得する。そして，それらの表現ベクトルに対して，候補となる単語群（語彙）の重み行列を掛けた線形変換と softmax 関数の確率分布の変換を施すことで，次にくる単語を予測しテキストを生成する。このプロセスによって結果的に特殊なトークンで置換された内容が復元され，それが正しい復元となるように各パラメータが最適化される。つまり，T5 の自己教師あり学習の狙いは，トークンで置換した欠落部分を単に予測するだけではなく，そのトークンを含むテキスト全体を生成してトークンの復元を実現することで，モデルに文脈理解能力とテキストの生成能力の両方を身につけさせることにある。T5 の事前学習におけるエンコーダとデコーダの処理のイメージを図 17（上）に示す。

この事前学習を大規模なデータセットと大規模なパラメータ数に基づいて実行することで高い汎化性能を得ている。なお T5 の事前学習で用いられるデータセットは，Google が作成した巨大データセット C4（Colossal Clean Crawled Corpus）というものであり，Web から収集されたデータにクリーニング処理をした 745GB のデータセットである。モデルのパラメータ数に関しては，たとえば T5-Base で 2.2 億，T5-11B で 110 億となっている。

T5 のファインチューニングでは，特定のタスク（翻訳，要約，分類，感情分析など）を指示するプレフィクス（prefix）というラベルを付けたデータを学習する。これにより，一つのモデルでさまざまなタスクに高い性能で対応することができる。たとえば英語をドイツ語に翻訳するタスクでは，プレフィクスとして“translate English to German：”というラベルを与える。つまり，T5 のファインチューニングで追加学習するデータは，「translate English to German：英語文」の入力文章と「ドイツ語文」の出力文章のペアとなる。このファインチューニングにより，

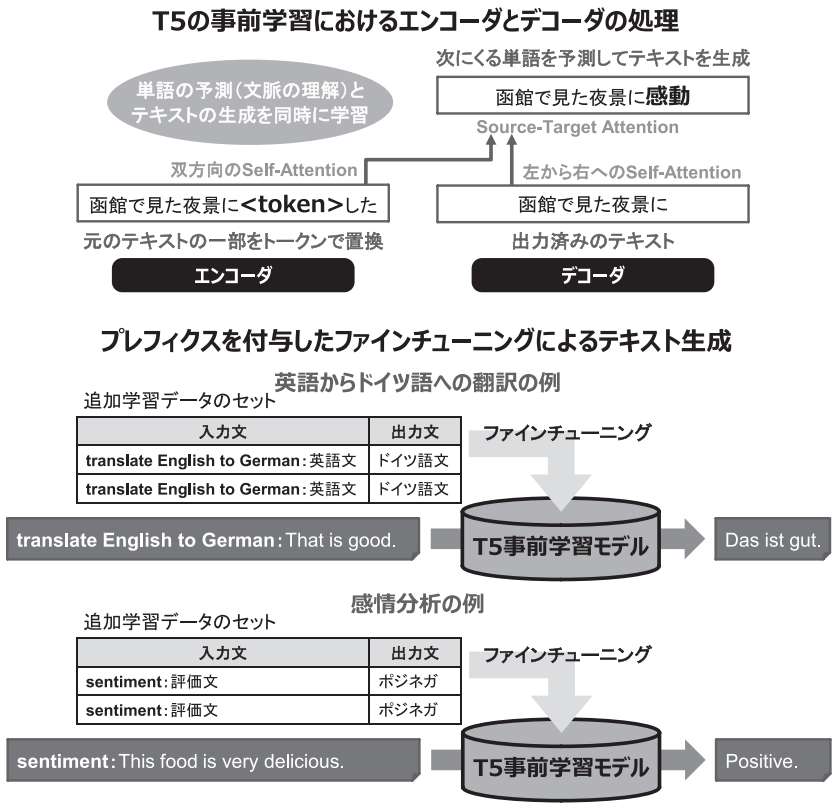


図 17　T5 の事前学習とテキスト生成

そのプレフィクスで指示されたタスクに対応したモデルに更新される。T5 のファインチューニングとタスク別のテキスト生成のイメージを図 17（下）に示す。

GPT でも Few-Shot Learning でさまざまなタスクに対応可能だが，これはそのタスクの例やヒントを与えているだけで，モデル自体は更新されない。そのため，GPT は汎用性はあるが特定のタスクに特化したモデルとはならず，さらに良い例となるショットの提供を含む適切なプロンプトの設計も求められる。また，GPT でもテキストで入力したタスクに対してその回答がテキストで出力され，テキストからテキストに変換されるという形式でいえば T5 と同様だが，GPT はあくまでも入力したテキストの次にくる単語を予測しており，T5 のように入力と出力の対応関係を強く捉えた生成ではないため，個別のタスクによっては性能が劣ることがある。

T5 はさまざまなタスクに応じた柔軟で高い性能のテキスト生成ができるというメリットがある一方で，計算資源の要件にはハードルもある。事前学習では膨大な計算資源が必要であり，ファインチューニングにおいても高い性能のモデルを実現するには比較的多くの計算資源が求められる。

5.　大規模言語モデルとテキストマイニング

ここまで解説したように，第 3 次 AI ブームは「Transformer」の登場により自然言語処理の分野で大きなブレイクスルーが起きた。そして，その Transformer のアーキテクチャをベースに膨大なテキストデータを学習させた汎用的な大規模言語モデルとして，「BERT」「GPT」「T5」などが誕生した。これらはテキストのデータを対象とした処理モデルであるが，一方でテキストデータの分析と活用といえば，「テキストマイニング」が広く利用されている。テキストマイニングは，本節第 2 項で取り上げた従来の自然言語処理技術をベースにしたデータマイニングのアプローチであるが，大規模言語モデルが登場したからといってテキストマイニングが古臭い手法となるわけではない。それは両者がどちらもテキストデータを扱う技術という点では共通するが，その特徴や用途には大きな違いがあるためである。ここでは大規模言語モデルとテキストマイニングの位置づけについて，両者の特徴を対比しながら解説する。

5.1　大規模言語モデル

まずは本節第 4 項で解説した大規模言語モデルについてもう一度整理する。「BERT」「GPT」「T5」は，Attention の仕組みだけを用いた「Transformer」というアーキテクチャに基づいて，大量のテキストデータから汎用的な言語特徴を学習した大規模言語モデルである。大規模言語モデルの種類と構成の違いを図 18 に示す。Transformer の構造のなかでも，BERT はエンコーダ，GPT はデコーダ，T5 はエンコーダとデコーダを利用したモデルである。

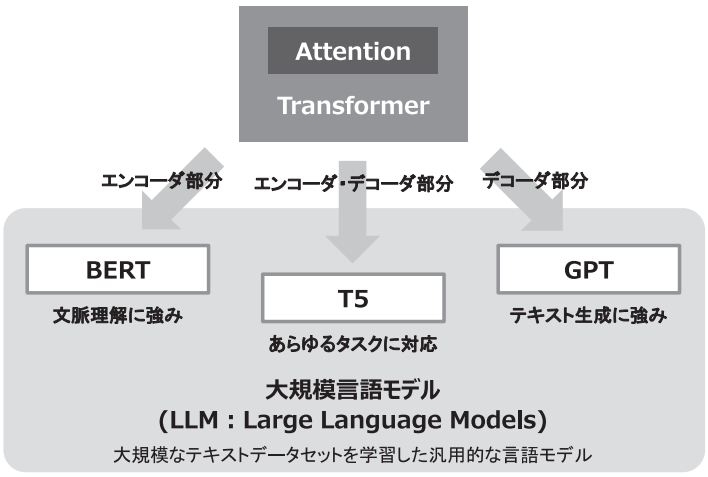


図 18　大規模言語モデルの種類と構成

BERT は文脈理解に強みがあり，文章分類や感情分析などのタスクに適しているといわれる。GPT はテキスト生成に強みがあり，文章生成や文章補完，要約作成，対話生成，言語翻訳などのタスクに適しているといわれる。T5 は文脈理解とテキスト生成の両方の能力を有していることであらゆるタスクに対応できる柔軟性があるといわれるが，比較的多くの計算資源が求められるといわれる。

2022 年 11 月に OpenAI は GPT を利用した AI チャットボットサービス "ChatGPT" をリリースしたが，これを機に世界中で生成 AI ブームが巻き起こり，大規模言語モデルのなかでは GPT が主流となったといえる。

5.2　テキストマイニング

テキストマイニングは大量のテキストデータから，その文書に含まれる単語を抽出し，単語をベースに文書に記述されている傾向を把握するデータマイニング手法である。テキストという定性的なデータでも定量的に集計したり統計的な分析を可能にしている。本節第 2 項でも解説したとおり，テキストマイニングは従来の自然言語処理技術を利用した手法であり，文書に含まれる単語を抽出してその品詞を割り当てる「形態素解析」と，その単語間の文法的な係り受け関係を抽出する「構文解析」を基本技術としている。

現在では多くのテキストマイニングツールが存在し，自然言語処理の専門知識がなくても，分かりやすいインタフェースで直感的な操作が可能で，テキストデータを活用したいビジネスの現場では人気のある分析ツールである。たとえば，無償のツールでは，立命館大学の樋口耕一教授が作成した KH Coder やユーザーローカル社の AI テキストマイニングなどが，有償のツールでは，NTT データ数理システム社の Text Mining Studio やプラスアルファ・コンサルティング社の見える化エンジンなどが知られている。

テキストマイニングのツールでよくある分析機能のアウトプット例を図 19 に示す。もっとも単純な分析機能は単語の頻度集計である（図 19 左）。形態素解析によって抽出された単語や，構文解析によって抽出された係り受け表現の出現頻度を集計しランキングすることで，分析対象としているテキストデータの文書情報では，どのような記述が多いのかという全体像を把握する。また，1 件の文章の中で同時に出現しやすい単語同士をネットワークでつなぐ共起ネットワークという分析機能がある（図 19 中央）。単語の頻度集計では，それぞれの単語が独立して集計されるため単語間のつながりが分からないが，共起ネットワークではつながりのある単語のかたまりが可視化される。こうしたネットワーク結果から，このテキストデータの文書情報ではどのような話題が形成されているのか考察できる。さらに少し高度な分析機能として，コレスポンデンス分析（あるいは数量化Ⅲ類）というものもある（図 19 右）。これは，テキストデータに紐づく属性情報（たとえばアンケートデータであれば回答者の性別・年代，特許データであれば出願年や出願人といった属性情報）と文書情報内の出現単語との対応関係を同じ平面上にマッピングしたものである。そのマップの属性と単語の位置関係からそれぞれの属性の記述傾向を把握できる。

このようにテキストマイニングは，テキストデータの文書情報に含まれる単語や係り受け表現を抽出することを基本とし，その単語や係り受けをベースに集計したり，属性情報とも絡めて統計解析を実行することで，テキストデータ全体の記述傾向を可視化し，現状を把握できる手法となる。図 19 では特許文書のテキストデータにテキストマイニングを実行した例を示しているが，たとえばこうした分析により近年の技術動向や競合他社の出願傾向を把握し，自社の研究開発の計画や M&A などの経営戦略に活かすこともできる。他にもビジネスにおけるテキストデータの活用例を挙げると，たとえば自由記述回答のあるアンケートのテキストデータを分析すれば，通常の選択式回答からは得られない回答者の潜在的なニーズを把握でき，マーケティング戦略に活かすことができる。口コミと呼ばれる Web 上のレビュー情報のテキストデータを分析すれば，消費者の評価を競合他社の商品と比較しながら理解し，商品企画に活かすこともできる。また，コールセンターに寄せられた問い合わせ内容が記録されたテキストデータを分析すれば，顧客の要望や不満を理解し，自社の商品やサービスの改善に活かすこともできる。

このように，テキストマイニングでテキストデータを分析し活用することは，ビジネスの現場の課題解決において価値の高い取り組みである。たとえ使用されている技術は従来の自然言語処理技術であっても，その分かりやすさからくる有用性は高いといえる。

頻度集計	共起ネットワーク	コレスポンデンス分析
単語や係り受け表現の出現頻度を集計して，どのような記述が多いのか，おおまかな全体像を把握する	同時に出現しやすい単語同士をネットワークでつなぎ，そのかたまりからどのような話題があるか考察する	属性情報と出現単語との対応関係を同じ平面上にマッピングし，その位置関係から属性の傾向を把握する

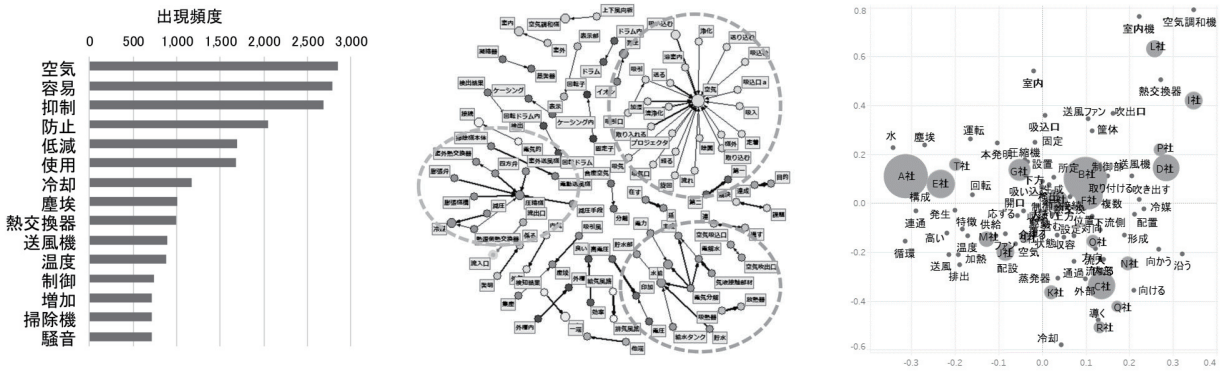


図 19　テキストマイニングのアウトプットの例

5.3　大規模言語モデルとテキストマイニングの比較

大規模言語モデルとテキストマイニングは，どちらもテキストデータを扱う手段という意味では共通するが，それぞれの特徴や用途には大きな違いがある。観点別に特徴の違いをまとめたものを表 1 に示す。ただし，この表は大規模言語モデルとテキストマイニングを対比することで，その違いをわかりやすく理解することを目的に筆者の見解でまとめたものである。タスクの条件や分析データの内容によって一概にはいえないため，あくまでも参考程度に捉えてほしい。以下にそれぞれの違いについて説明する。

表 1　大規模言語モデルとテキストマイニングの特徴の対比

対比の観点	大規模言語モデル	テキストマイニング
用途	**文脈の評価や文章の生成**	**特定のテキストデータの特徴の把握**
活用例・分析例	文章分類，感情分析，文章生成，要約作成，質問応答，対話生成，言語翻訳	単語頻度の集計・推移，単語のネットワーク・マップ化，属性別の特徴語把握
課題解決の方式	**直接的** （アウトプットそのものが課題解決において直接的な価値を提供）	**間接的** （アウトプットは課題解決のための意思決定に資する価値を提供）
アウトプットの軸	**推論ベース** （事前学習した汎用モデルで推論）	**事実ベース** （特定のデータの情報のみで表現）
データ処理の軸	**文脈** （単語間の相互依存性）	**単語** （特定の単語の出現有無）
適用範囲の性質	**汎用性** （言語の汎用的な特徴を事前学習し多様なタスクに適用可能）	**個別性** （特定のデータセットに存在する個別の特徴や傾向を把握可能）
結果の形式	**定性的** （テキスト形式による出力）	**定量的** （統計解析による集計と可視化）
入力データ規模	**限定的なテキストデータ** （入力単語数に制限あり）	**大量のテキストデータ** （入力単語数に制限なし）

まずは用途の違いについて確認すると，大規模言語モデルは大規模なテキストデータセットを事前学習した汎用的な言語モデルであり，その学習された言語の汎用的な特徴に基づいて，文章の文脈を評価したり，新しい文章を生成したりできる。これにより，文章分類や感情分析，文章や要約の作成，質問応答や対話生成，言語翻訳などを得意タスクとしている。一方，テキストマイニングは手元にある特定のテキストデータの特徴を把握し，そこから有用なインサイトを得ることを目的とした手法である。特定のテキストデータに出現する単語を抽出し，その単語の頻度情報をベースとした集計や定量的な統計解析を実行することで，図 19 で示したような可視化を行う。

ビジネスの課題解決におけるテキストデータの活用という命題の下では，大規模言語モデルは文章要約や言語翻訳など，そのアウトプットそのものが課題解決において直接的な価値を提供しているといえる。一方，テキストマイニングは，そのアウトプットは課題解決における意思決定に資する価値を提供するものであり，大規模言語モデルと対比すると，間接的なアプローチで課題解決に貢献していると捉えることができる。

それぞれのアウトプットが何に基づいて形成されるのかという点では，大規模言語モデルは事前学習した汎用的な言語モデルに基づいて推論された結果であるのに対し，テキストマイニングは特定のテキストデータの情報，とりわけ出現単語の頻度に基づいて可視化された結果である。つまり，大規模言語モデルは推論ベースに，テキストマイニングは事実ベースにアウトプットが形成されると捉えることができる。たとえば大規模言語モデルを用いて文章分類や文章生成，言語翻訳などを行う場合，その結果は推論に基づくものであるため誤った結果となる可能性もある。また ChatGPT の利用においても，入力するプロンプトによって生成される結果は大きく影響を受け，同じプロンプトであっても異なる結果が生成されることもある。一方，テキストマイニングは特定のテキストデータにどの単語が何件出現するのかなど，そのデータの特徴をそのまま反映しており，結果は事実そのものである。正解や不正解といった概念はなく，当然同じ条件の分析において結果が変動することもない。

次に，得意な処理と不得意な処理について確認しながらそれぞれの特徴を比較する。まずは大規模言語モデルが得意で，テキストマイニングが不得意な処理についてである。その代表として，大規模言語モデルは文脈を理解することが得意であるが，テキストマイニングは得意ではないということが挙げられる。たとえば「監督は選手にサインを送った」と「彼は契約書にサインした」という 2 つの文章で使われる「サイン」は同じ単語だが，違う文脈で使用されている。テキストマイニングではどちらも同じ「サイン」として認識し，文脈による区別ができないが，大規模言語モデルではこれらを違う文脈で登場する単語として認識できる。他を挙げると，テキストマイニングには文章のポジティブ・ネガティブを判定する感情分析を行う機能もあるが，これは固定化された語彙リストやルールに基づいて評価するものであり，文脈を捉えた評価になっていない。これに対して大規模言語モデルは文脈を理解した感情分析を実行することができる。こうした違いは，テキストマイニングでは特定の単語の出現有無をデータ処理の軸としているが，大規模言語モデルでは単語の順番や単語間の相互依存性といった文脈を考慮した単語の特徴ベクトルをデータ処理の軸としている違いで現れる。

逆に大規模言語モデルが不得意で，テキストマイニングが得意な処理もある。たとえば，大規模言語モデルは大規模なテキストデータセットを事前学習した汎用性のある特徴を捉えたモデルだが，個別性の強い特徴を捉えることは得意とはいえない。これは過学習をしないように汎用的なモデルの学習を進めているため当然であるが，テキストマイニングでは手元にある特定のテキストデータを処理し，そのデータ特有の個別の特徴や傾向を可視化できる。また，大規模言語モデルのアウトプットは文章生成や質問応答など，その形式は定性的な情報だが，テキストマイニングのように定量的な統計解析を行うことは得意ではない。他に，大規模言語モデルが不得意とする処理として，入力できるデータの規模がある。大規模言語モデルは事前学習時にはとてつもない規模のテキストデータを学習するが，それを再利用するときに入力するテキストデータの規模は限定され，テキストマイニングのように大量のテキストデータセットを一気に処理することには向いていない。これは大規模言語モデルでは一度に処理できる単語数に制約があるためである。正確には単語ではなく，トークンと呼ばれる単位で処理をしているが，大規模言語モデルでは計算コストの爆発抑制のため，このトークンの数に上限を設けている。たとえば，BERT の上限トークン数は 512 個，GPT-3 の上限トークン数は 4096 個である。生成 AI ブームによって大規模言語モデルの開発の競争が激化する中で，この上限トークン数の拡大も進んでいるが，無制限ということはない。なお，英語の場合は単語とトークンが一致するこ

とが多いが，日本語の場合は，単語の数の方がトークンの数よりも少なくなることが一般的である。

大規模言語モデルの開発が飛躍的に加速するなか，ここで挙げた大規模言語モデルの不得意な処理は克服される流れにあるが，技術の本質として大規模言語モデルとテキストマイニングにはどのような特徴がありどのような違いがあるのか理解することは重要である。

5.4　大規模言語モデルとテキストマイニングの組み合わせ

大規模言語モデルとテキストマイニングは，こうした違いがある一方で，大規模言語モデルとテキストマイニングを相互補完的に組み合わせて活用することで，より有用な分析アプローチを形成できる。ここでは大規模言語モデルをテキストマイニングの前段階に活用するパターンと，テキストマイニングを大規模言語モデルの前段階に活用するパターンについて考える。

（1）大規模言語モデルをテキストマイニングの前段階に活用するパターン

たとえば，テキストマイニングによる感情分析は精度があまり高くないことがいわれるため，先に大規模言語モデルで文脈に基づいた高い精度の感情分析を施し，各感情別の特徴をテキストマイニングで単語ベースに把握するということが可能である。またこれと類似するが，BERT を用いて文章を分類し，その分類された文章群ごとにテキストマイニングを実行すれば，各分類の特徴の違いを単語ベースに理解することができる。なお，BERT の文章分類は，事前に分類ラベル付きのテキストデータでモデルをファインチューニングし，それを用いて分析用のテキストデータを分類するという教師あり分類ができる。さらに，BERT は文脈に基づいた文章の特徴ベクトルをエンコードできるため，分析用のテキストデータに対してベクトル表現をエンコードし，それに基づいた次元圧縮やクラスター分析によって教師なし分類もできる。

また，テキストデータ全体をそのままテキストマイニングするのではなく，GPT を用いてそのテキストデータからテーマ別に要約を生成し，そのテーマごとに生成された要約文をテキストマイニングすれば，全体では見えてこなかったテーマ別の特徴を把握できる。たとえば，特許文書のデータを分析する場合，特許文書全体に記載されている技術の内容と，用途の内容を分けてその要約を生成し，それぞれの特徴をテキストマイニングで把握する。これにより，特許の技術と用途を区別した分析や両者の関連性を見る分析が可能となる。

他には，多言語のテキストデータをまとめてテキストマイニングしたいときに，GPT や T5 で同一言語に翻訳してからテキストマイニングを実行したり，テキストマイニングの辞書作りにおいて，GPT を使って類義語の候補を生成するといった活用アイデアも考えられる。ただし先述のとおり，大規模言語モデルは一度に大量のテキストデータを処理することは得意ではないため，分析対象のテキストデータの量が多いときには，事前に分割してから入力するなどの工夫が必要である。

（2）テキストマイニングを大規模言語モデルの前段階に活用するパターン

たとえば，まずは従来通りのテキストマイニングを実行することで，特徴を可視化したり文章のクラスタリングをする。そして，その結果に基づいてフィルタリングしたデータに対して大規模言語モデルを適用することが考えられる。これにより，テキストマイニングの結果でさらに深掘りして注目したい対象が見つかった場合，その対象に絞り込んだテキストデータに対して，大規模言語モデルで感情分析や要約生成などを行えば，その対象の理解をより深めることができる。特にテキストマイニングで可視化された定量的な特徴は，どのような内容を含んでいる特徴なのか定性的に理解できる。また，絞り込まれたテキストデータに対して大規模言語モデルで文章分類を行えば，注目した特徴をさらに細分化して分類でき，各分類の細かい傾向を理解できる可能性もある。

なお，テキストマイニングを大規模言語モデルの前段階として活用する場合は，最初から大量のテキストデータを対象とすることができる。

おわりに

本節では，近年の自然言語処理技術について，その全体像を整理して俯瞰した理解をすることを目的に，①従来の自然言語処理技術，②トピックモデル，③深層学習モデルという３つに大別して代表的な手法を取り上げ，その概

要と特徴について，また互いの手法の関係性について解説した。本節で強調したいことは，どれが新しい手法でどれが古い手法であるとか，どの手法がどの手法より優っているのか劣っているのかということではなく，それぞれの手法において固有の特徴があるということである。そしてその特徴を理解した上で，今解決したい課題にはどの手法が有効であるのか見極め選択することが重要である。

昨今の生成 AI ブームでは，あらゆる課題が GPT に投げかければ解決できると考えている人も少なくない。今一度人間と GPT の違いを認識しておくと，GPT は人間のように内部にある知識を引き出しているわけではなく，また投げかけられた入力に対して思考しているわけでもない。まるで会話のように感じる流暢なやりとりや，その出力内容の質の高さから，そのように錯覚してしまいそうだが，GPT は大量のテキストデータから汎用的な言語パターンを見つけ，与えられたテキストの次にくる単語を予測していることが技術の核心である。つまり，○○を教えてと質問して有益な回答が得られても，その知識が元々 GPT のモデルの内部に存在していて，質問によってそれが引き出されたというわけではない。ただ，大規模なテキストデータを大規模なパラメータ数で学習すると，与えられたテキストの次にくる単語を予測するという単純なシステムであっても，まるで大量の知識を有しているように，そしてまるで人間と会話しているように感じられるモデルが出来上がってしまうということである。もはやそれがなぜ実現できるのかは分からず，ある意味恐ろしい技術であるが，その賢すぎる AI に人類が熱狂してしまうことも納得ができる。

昨今の GPT 一辺倒となる流れは，Transformer や BERT が発表された直後も同様の現象が見られた。GPT は BERT の少し前に発表されたが，ChatGPT が登場するまでは，とりあえず BERT を適用したという論文や，BERT の亜種を提案する論文が溢れていた。それが ChatGPT の登場後は GPT ベースの大規模言語モデル一色となった。自身が解決したい課題に対して，話題の技術や一番新しい技術が常に最適であるというわけではない。これはたとえばパソコンや家電を購入するときに，その知識がない人が使いもしないような高機能の一番高くて一番良いものを求めようとする行動に類似している。本来あるべき購入の姿は，このような用途で使用するからこのような機能が欲しくて，それならばこの製品が最適であると選択できることである。テキストデータの分析と活用においても，課題に応じて適切な手法や技術を選択できることが重要である。たとえば形態素解析や Bag-of-Words は昔からある古い手法で，テキストの文脈は考慮できていないが，単語とその頻度で分析するそのアプローチは非常にシンプルで分かりやすく，それをベースにしたテキストマイニングはビジネスの現場で長く愛用されている。トピックモデルでも，PLSA は観測データに完全に依存し過学習がされ，一方 LDA では過学習を抑制したトピックの推定ができるとされているが，過学習をしている PLSA の方が観測データの固有の特徴の再現力は高いと捉えることもできる。大規模言語モデルも汎用性の高さはあるが，ファインチューニングで追加学習するデータや，与えるプロンプトによって結果が変動する。またなぜそのような変動が現れるのかということを追跡することも困難であるため，ケースによっては扱いにくい手段となることもある。

本節では大規模言語モデルとテキストマイニングの違いについても述べたが，生成 AI の進歩は確かに凄まじいため，プロンプトによっては生成 AI だけでテキストマイニングと同等の機能を実行する命令ができる可能性もある。そうすると，進化した生成 AI があればこうした手法の違いの区別は関係なく，すべて対応できるという考え方もあるかもしれない。しかしそうであったとしても，人間が課題解決を進めるにあたって，どのような特徴を有する手段があるのか理解し，どの手段がその課題解決に有効であるのか選択できることが重要であるということは変わらない。したがって，テキストデータの分析や活用で解決したい課題に応じて適切な手法を選択できる，あるいは複数の手法を組み合わせた適切な設計ができるためには，まずは自然言語処理技術の各種手法の特徴を冷静に俯瞰し，それぞれの仕組みや機能の本質を理解することが肝要である。本節の解説がそうした理解の一助になれば幸いである。

文　　献

1）金明哲：『テキストアナリティクス』，共立出版（2018）

2）S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman : “Indexing by Latent Semantic Analysis”, Journal of the American Society for Information Science, Vol. 41, No. 6, pp. 391-407 (1990)

3）D. D. Lee and H. S. Seung : “Learning the parts of objects by non-negative matrix factorization”, Nature, vol. 401, pp. 788-791 (1999)
4）T. Hofmann : “Probabilistic Latent Semantic Analysis”, Proc. of the 15th Conference on Uncertainty in Artificial Intelligence, pp. 289-296 (1999)
5）C. Ding, T. Li, and W. Peng ; “Nonnegative Matrix Factorization and Probabilistic Latent Semantic Indexing: Equivalence, Chi-square Statistic, and a Hybrid Method”, Proc. of the 21st National Conference on Artificial Intelligence, pp. 342-347 (2006)
6）D. M. Blei, A. Y. Ng, and M. I. Jordan : “Latent Dirichlet Allocation”, Journal of Machine Learning Research, Vol. 3, pp. 993-1022 (2003)
7）D. E. Rumelhart, G. E. Hinton, and R. J. Williams : “Learning representations by back-propagating errors”, Nature, Vol. 323, No. 6088, pp. 533-536 (1986)
8）G. E. Hinton, S. Osindero, and Y-W Teh : “A fast learning algorithm for deep belief nets”, Neural Computation, Vol. 18, No. 7, pp. 1527-1554 (2006)
9）G. E. Hinton, and R. Salakhutdinov : “Reducing the Dimensionality of Data with Neural Networks”, Science, Vol. 313, No. 5786, pp. 504-507 (2006)
10）A. Krizhevsky, I. Sutskever, and G. E. Hinton : “ImageNet Classification with Deep Convolutional Neural Networks”, Proc. of the 25th International Conference on Neural Information Processing Systems (NIPS 2012), pp. 1097-1105 (2012)
11）Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner: “Gradient-based learning applied to document recognition”, Proc. of the IEEE, Vol.86, No.11, pp. 2278-2324 (1998)
12）T. Mikolov, K. Chen, G. Corrado, and J Dean : “Efficient Estimation of Word Representations in Vector Space”, Proc. of the International Conference on Learning Representations (ICLR) (2013)
13）Q. V. Le, and T. Mikolov : “Distributed Representations of Sentences and Documents”, Proc. of the 31st International Conference on Machine Learning (ICML-14) (2014)
14）J. J. Hopfield : “Neural networks and physical systems with emergent collective computational abilities”, Proc. of the National Academy of Sciences (PNAS), Vol. 79, No. 8, pp. 2554-2558 (1982)
15）M. I. Joradn : “Serial Order: A Parallel Distributed Processing Approach”, ICS Report 8604 (1986)
16）J. L. Elman: “Finding structure in time”, Cognitive Science, Vol. 14, No. 2, pp. 179-211 (1990)
17）T. Mikolov, M. Karafiat, L. Burget, J. Cernocky, and S. Khudanpur : “Recurrent neural network based language model”, ISCA Interspeech, pp. 1045-1048 (2010)
18）Y. Bengio, P. Simard, P. Frasconi : “Learning Long-Term Dependencies with Gradient Descent is Difficult”, IEEE Transactions on Neural Networks, Vol. 5, No. 2, pp. 157-166 (1994)
19）S. Hochreiter, and J. Schmidhuber : “Long Short-Term Memory”, Neural Computation, Vol. 9, No. 8, pp.1735-1780 (1997)
20）杉山聡：『本質を捉えたデータ分析のための分析モデル入門』，ソシム（2022）
21）佐藤大輔，和知徳磨，湯浅晃，片岡紘平，野村雄司：『 BERT 入門 プロ集団に学ぶ新世代の自然言語処理』，リックテレコム（2022）
22）I. Sutskever, O. Vinyals, and Q. V. Le : “Sequence to sequence learning with neural networks”, Proc. of the 27th International Conference on Neural Information Processing Systems (NIPS 2014), pp. 3104-3112 (2014)
23）D. Bahdanau, K. Cho, and Y. Bengio : “Neural machine translation by jointly learning to align and translate”, arXiv preprint arXiv : 1409. 0473 (2014)
24）A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin : “Attention is All You Need”, Proc. of the 30th International Conference on Neural Information Processing Systems (NeurlPS

2017), pp. 6000-6010 (2017)

25) A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby : "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", arXiv preprint arXiv : 2010. 11929 (2020)

26) A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever : "Zero-Shot Text-to-Image Generation", arXiv preprint arXiv : 2102. 12092 (2021)

27) J. Devlin, M. W. Chang, K. Lee, and K. Toutanova : "BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding", arXiv preprint arXiv : 1810. 04805 (2018)

28) J. Devlin, M. W. Chang, K. Lee, and K. Toutanova : "BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding", Proc. of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NCCAL-HLT 2019), Vol. 1, pp. 4171-4186 (2019)

29) A. Radford, K. Narasimhan, T. Salimans, and I. Sutskeve : "Improving Language Understanding by Generative Pre-Training", Open AI technical report (2018)

30) A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever : "Language Models are Unsupervised Multitask Learners", Open AI technical report (2019)

31) T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei : "Language Models are Few-Shot Learners", arXiv preprint arXiv : 2005. 14165 (2020)

32) C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu : "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer", arXiv preprint arXiv : 1910. 10683 (2019)