16 量子化

量子化とは、AIモデルのパラメータを少ないビット数で表現して、低い精度で計算する技術です。低精度と言うと悪いことのように聞こえるでしょうが、メモリ使用量の削減と計算の高速化につながる生成AIの時代には欠かせない技術です。

モデルサイズとGPUのVRAMの関係

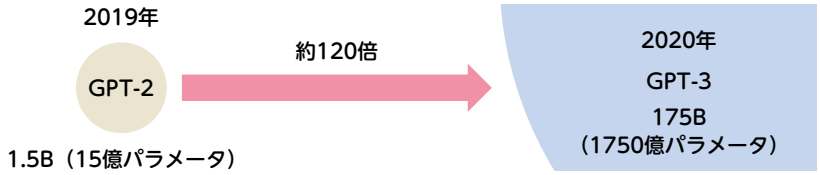
ニューラルネットワークの節（p.067参照）で解説した通り、深層学習の計算の大部分は次のような形をしています。ここでは説明のためにaやxの個数は4個で済ませていますが、実際には千や万の単位で並びます。

$$y = a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4$$

x_1, x_2, x_3, x_4は関数への入力です。1つ前の層の出力を受け取るので、さまざまな値を取る可能性があります。そこでx_1, x_2, x_3, x_4は通常FP32やFP16などの表現力が高い形式が使われます。

a_1, a_2, a_3, a_4はモデルのパラメータです。これらを求めることが目的である学習ではその値は決まっていませんが、推論のときは学習時に決まった値のまま変化しません。これらももともとはFP32やFP16などで表されていましたが、大規模言語モデルのモデルサイズがあまりにも増えてしまったため、パラメータを表現するのに16ビットでも多すぎるようになってきました。

■1年でモデルが100倍以上に巨大化



例えば2019年に発表されたGPT-2は1.5B（15億個）のパラメータを持ち、FP16（1パラメータあたり16ビット）で表現すると3GBになります。一方、その1年後に発表された後継のGPT-3は175B（1750億パラメータ）とGPT-2の100倍以上になりました。これをFP16で表現すると350GBになります。

GPUで高速に計算するには、データをVRAM（GPUのメモリ）にすべて載せる必要があります。VRAMが24GBのGeForce RTX4090（NVIDIAの一般向けの最上位GPU）で考えると、GPT-2は1枚のRTX4090に楽々載りますが、GPT-3はRTX4090が15枚必要です。これはモデルをメモリに載せるだけで必要な量なので、計算に必要な量を加えるとさらに増えてしまいます。なお、ChatGPTに使われるGPT-3.5は倍の350B（3500億パラメータ）です。

■GPT-2からGPT-3.5を表すのに必要なメモリ量

	32ビット	16ビット	8ビット	4ビット
GPT-2（1.5B）	6GB	3GB	1.5GB	0.75GB
GPT-3（175B）	700GB	350GB	175GB	87.5GB
GPT-3.5（350B）	1400GB	700GB	350GB	175GB

量子化

もし数値を8ビットや4ビットといったより少ないビット数で表すことができれば、その分だけメモリサイズが減って、GPUの数も減らすことができます。そのような技術の1つに**量子化**があります。

先ほどの関数で、具体的に量子化手法について考えてみましょう。このa_1, a_2, a_3, a_4たちは「膨大に多いが、推論時は変化しない」という特徴があるので、それをうまく使います。

$$y = a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4$$

試しに3ビットで表す方法を考えてみましょう。一番単純な方法は3ビットの整数にしてしまうことです。しかし、モデルのパラメータは0に近い値を取ることが多く、そのまま整数に丸めると、全部0になってしまいかねません。

17 GPUを使った深層学習

深層学習、特に大規模言語モデルの学習や推論にはGPUが欠かせないと言われています。本節では、もともとグラフィックス専用に開発されたGPUがどのようにして深層学習に必要不可欠な存在になったかを見ていきます。

計算を速くする方法

「百ます計算」という計算練習方法があります[1]。縦横10マスの左と上に数が書いてあり、交差するところに2つの数を足したり掛けたりした答えを書いていく方法です[2]。

■百ます計算

×	3	9	1	7	2	6	8	4	0
6									
9									
8									
5									
1									
7						42			
2									
0									
3									

百ます計算（掛け算）
縦と横の交差したマスに
2つの数を掛け算した数を答える

計算を速くするには百ます計算で練習をしましょう、という話ではありません。1回の計算時間は変わらないまま、この表すべてをできるだけ短時間で埋めるにはどうすればいいか、という問題を考えてみましょう。

普通に考えれば100回計算が必要ですが、2人で半分ずつ分担すれば50回の時間で終わります。人数を増やせば時間はどんどん短くできて、100人で1人1マスずつ担当すればなんと1回分の計算時間で終わります[3]。

実はGPUの高速計算はまさにこの方法で実現しています。GPUは大量に持っている演算器という部品を、計算が必要なところに1つずつ割り当てます。そうすることで10×10マスどころか、100×100のマスでも1回分の計算時間で済ませられます。

GPU vs CPU

あらためて、**GPU**とは "Graphics Processing Unit" の略で、コンピュータのグラフィックスを処理する部品です。3Dゲームなどの高度なグラフィックス処理に必須で、SONY PlayStationやNintendo Switchなどのゲーム機にも搭載されています。通常のパソコンではGPUはCPUと1つのチップに統合されており、OSやビジネスアプリケーションの描画を高速化します。また、動画のエンコード・デコード（圧縮・展開）でもGPUは活躍します。

映像に関係したタスクだけでなく、GPUは計算が高速という特徴があります。以前はコンピュータにおける計算のほぼすべてをCPUで行っていましたが[4]、現在は深層学習やAIの計算は主にGPUで行われています。

GPUの計算が高速だからといって、CPUの役割をすべてGPUに置き換えることはできませんし、今後もCPUがなくなることは考えられません。実は、GPUは特定の条件を満たす計算だけが高速なのです。GPUの得意な計算を明らかにするため、次のような実験をしてみましょう。

- **ランダムな数値のリストを用意し、そのリストの値1つ1つの2乗を計算**
- **CPUとGPUそれぞれで、その計算にかかった実行時間を計測**
- **リストの大きさをさまざまに変えて、実行時間がどのように推移するかを調べる**

[1] 百ます計算 - Wikipedia　https://ja.wikipedia.org/wiki/百ます計算

[2] ちなみに掛け算の百ます計算は、ベクトルで言えば外積（outer product）、行列で言えば10×1行列と1×10行列の行列積に相当し、LoRA（p.184参照）で用いられる低ランク近似（ランク1）も同じ計算です。

[3] ただし100人が1枚の紙に書き込む時間は考慮していません。現実のGPUを使った計算でも、バス（計算結果をCPUに渡したりする通信路）がボトルネックになることは往々にしてあります。

[4] かつては計算を専門に行うFPU（Floating Point Unit）をCPU外に設置していた時期もありましたが、現在はその機能はCPU内に組み込まれています。

21 トークナイザー

トークナイザーは、テキストをコンピュータで扱いやすい単位に分割するためのツールです。この節では、トークナイザーの役割や学習方法について解説します。

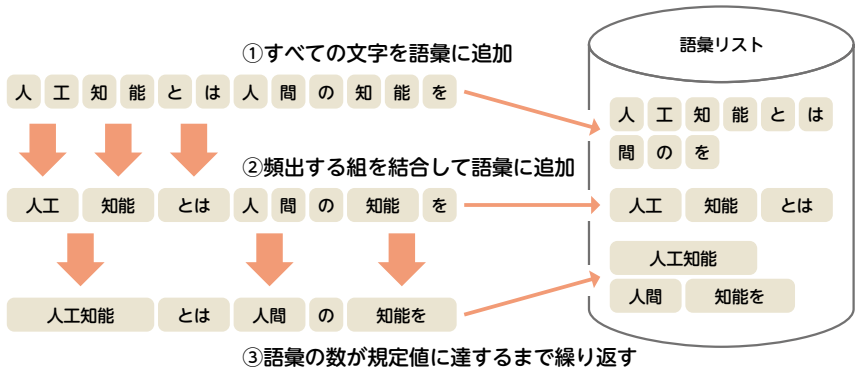
トークナイザーの学習

トークナイザーとは、テキストをコンピュータに扱える単位（トークン）に分割するツールです。分割だけではなく、トークンをID番号に変換したり、逆にトークンIDの列からテキストを復元するのもトークナイザーの役割です。

現在は文字や単語ではなく統計的なサブワード分割が主流であり、どのようなサブワード分割が良いかをテキストから決定するプロセスがトークナイザーの学習です。

トークナイザーを学習するには、必要な言語やドメインに合った十分な量のテキストデータを用意します。次に、このテキストからアルゴリズムに従ってトークンを取り出し、語彙に追加していきます。ここでは、よく使われるByte-Pair Encoding（BPE）アルゴリズムによる学習手順を紹介します。

■ トークナイザーの学習



まずは学習テキストを1文字ずつに分解し、それらをすべて語彙リストに登録します。次に、図の「知」と「能」のようにテキストの中で連続する頻度が多い語彙を結合し、新たな語彙として追加します。これを最初に決めた語彙数になるまで繰り返します。他の学習アルゴリズムも手順こそ違いますが、頻出表現を少ないトークン数で表せるようにする点は同じです。

■ トークナイザーの学習モデル

トークナイザー	概要
Byte-Pair Encoding（BPE）	テキストに頻出するサブワードの組をマージして新しいサブワードを作成することを繰り返す。
Unigram	文を生成する確率が大きくなるようにサブワードを追加。
WordPiece	単語の頻度とサブワードの頻度の積が最大になるようにサブワードを選択。

トークナイザーの学習は、最初に与える最大語彙数で制御します[1]。一般に語彙数が多いほど、同じ文章を表すのに必要なトークン数は少なくなります。

語彙数の違いによる影響を見てみましょう。オープンソースのトークナイザーSentencePiece[2]で日本語Wikipediaの本文テキスト全体[3]を語彙数5000と10000のそれぞれに対し学習し、日本国憲法の前文（646字）をトークン分割したときのトークン数と冒頭の分割の様子を表にしました（次ページ）。

日本国憲法とWikipediaの文章は似ていないので、分割の効率的には不利ですが、それでも語彙数を増やすとトークン数が減ることがわかります。分割の様子を見ると、語彙数5000では「国民」や「国会」などが2個のトークンに分かれているのに対し、10000では1個のトークンに結合しています。

[1] そのほかに語彙の最大文字数なども指定できます。

[2] SentencePiece: Unsupervised text tokenizer for Neural Network-based text generation. https://github.com/google/sentencepiece

[3] WikiExtractor　https://github.com/attardi/wikiextractorで作成した日本語Wikipediaの本文テキスト全体は約3.7GBになります（2024年現在）。

26 ニューラルネットワークの汎用性と基盤モデル

主に1タスク＝1モデルである通常の機械学習と異なり、ニューラルネットワークは特徴抽出など、タスクを横断する汎用的な能力が注目されてきました。そうした性質が基盤モデルの概念につながっています。

ニューラルネットワークによる特徴抽出

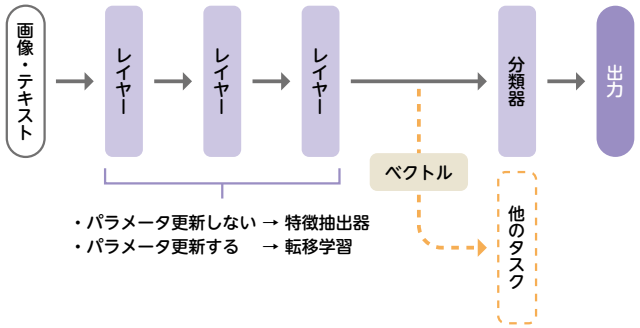
コンピュータが画像やテキストを理解するのは、実は簡単なことではありません。例えば、画像の1ピクセル（画素）だけを見て、あるいは文章の1文字だけ見て、その内容や意味を理解するのは難しいです。

そこで、データの中からコンピュータが意味のある処理ができる重要な情報を取り出すことが行われてきました。そのような情報を**特徴**と言います[1]。そして高い精度の実現には、データやタスクに合わせて特徴の抽出方法を設計する必要がありました。同じ画像でも、一般的な写真の分類に適した特徴が、レントゲン画像から病気の可能性を予測するのに適しているとは直感的には考えにくいですよね。しかしそれはとても困難でした[2]。

一方、ニューラルネットワークは基本的には画像やテキストをそのまま入力します。そして例えば画像分類をニューラルネットワークで解く場合、次ページの図のように、手前の層は画像の生データから特徴となるベクトルを抽出し、ネットワークの出口にあるシンプルな分類器に入力していると考えることができます。深層学習の発展により精度が上がってくると、ニューラルネットワークはそのまま優秀な特徴抽出器として、他のタスクに利用されるようになりました。

また、学習したモデルを他のタスク用に再学習して転用することを**転移学習**と言います。転移学習によって、先ほど直感的には考えにくいと言った「一般的な写真の分類モデルを、レントゲン画像からの病気予測に適応」みたいな畑違いのタスク横断もうまくいくことが報告されています[3]。

■ ニューラルネットワークによる特徴抽出と転移学習



基盤モデル

従来の機械学習は、タスク専用モデルをタスク専用データセットで学習するのが一般的でした。例えば自然言語処理の機械翻訳タスクでは、対訳コーパスと呼ばれる機械翻訳用のデータセット（例：英語とフランス語の同じ内容の文章の組）で、機械翻訳用のモデルを学習しました。文書分類タスクでは、文書分類用のデータセット（例：映画のレビューの文章と、内容が肯定的か否定的かを表すラベル）で、同じく文書分類用のモデルを学習しました。

一方、前項で紹介した特徴抽出や転移学習のように、ニューラルネットワークにはタスクを横断して汎用的に問題に適応できる能力があることがわかってきました。それをさらに進めた考え方が**基盤モデル**です。基盤モデルは転移学習を前提とした汎用モデルであり、大規模なデータセットで汎用的なモデルを学習し、それをタスクごとにチューニングします。基盤モデルのアプローチは、BERT（p.218参照）によって一気に普及しました。

[1] または数値化した特徴を指して「特徴量」とも言います。一方、日本の自然言語処理ではこれを「素性（そせい）」と呼びますが、英語ではどちらも"feature"になります。

[2] 優れた特徴抽出器は特許で保護されており、商用利用にはライセンス料が必要でした。

[3] Kim, Hee E., et al. "Transfer learning for medical image classification: a literature review." BMC medical imaging 22.1 (2022) : 69.

35 大規模言語モデルの学習 ～ファインチューニング～

基盤モデルの微調整の方法には、ファインチューニングやコンテキスト内学習（p.188参照）などがありますが、本節ではファインチューニングを含めた、モデルのパラメータ更新を伴う微調整方法について解説します。

ファインチューニング

大規模言語モデルの学習プロセスは大きく2段階に分かれます。第1段階の事前学習では、タスクによらない大量のデータを使って汎用的な**基盤モデル**を学習します（p.137参照）。この基盤モデルに対し、利用目的に合わせた**ファインチューニング**（「微調整」と訳される）を行うのが第2段階になります。

ファインチューニングの最も代表的な目的は、翻訳や要約、チャット（対話）などのタスクへの対応能力を上げる**インストラクション・チューニング**（指示に従うチューニング）です。他に文体や絵柄などのスタイル変更を目的とすることもあります。

また、モデルに知識を追加するためにファインチューニングが試みられることもありますが、後述する理由でうまくいかない可能性が高いです（p.190参照）。外部知識への対応はRAG（p.255参照）を用いましょう。

インストラクション・チューニングの効果の検証として、事前学習しただけのベースモデルとそれをファインチューニングしたモデルで[1]、質問の回答がどのように変わるかを見てみましょう（次ページの表）。

このように、大規模言語モデルはインストラクション・チューニングの有無で振る舞いが大きく変わります。基本的に事前学習は言語の知識を獲得することが目的であり（p.175参照）、会話や指示に応えることや明示的な文の終わりを学習していないため、この例のような冗長な連想ゲームのような出力になりがちです。一方、小規模なデータでもインストラクション・チューニングを行うと、回答が簡潔で的確になります。

Hugging Faceに公開されているベースモデル（基盤モデル）をチューニングしないで使いたい場合は、インストラクション・チューニング済みのモデル（モデル名に"-instruct"や"-chat"などが付いているもの）を選びましょう。

[1] Meta Llama 3 8Bを日本語で継続事前学習したrinnaのLlama 3 Youko 8Bモデル　https://huggingface.co/rinna/llama-3-youko-8bをベースモデルとし、これをllm-jp/databricks-dolly-15k-jaで1エポックQLoRA学習したモデルの比較。

■インストラクション・チューニングBefore-After

質問	ベースモデル	インストラクション・チューニング後
富士山の高さは？	富士登山は、富山県の富山市にある立山連峰の雄山に登ることを指す場合もある。立川市にある高尾山とは、混同しないように注意されたい。	3776メートル。富士は日本で最も高い山である。日本の最高峰は、富山県と長野県の境にある北アルプスの北岳である。
イギリスの首都は？	1. ロンドン 2. パリ 3. ベルリン 4. モスクワ 5. ワシントンD.C. 6. ニューヨーク 7. 北京 8. 東京 9. ソウル 10. 上海 11. バンコク 12. ムンバイ 13. カイロ 14. イスタンブール 15. ドバイ 16. アブダビ	イングランドのロンドンは、イギリスの国都である。ローマ帝国の時代に始まったこの都市は、その後何世紀にもわたって発展を続け、現在では世界で最も人口の多い都市のひとつとなっている。
The early bird catches the worm. を日本語に訳して	例文帳に追加 早起きは三文の徳だ . - Tanaka Corpus The early worm gets the bird. - Weblio Email 例文集 【例】早寝早起き .《諺》早寝は早朝の鳥を捕らえる . 既に死んでいる鳥は捕まえられない .	早起きは三文の徳。早起きな人は、虫を捕まえる。これは、早く起きた人は良いことをするという意味である。また、朝早い人は仕事を始める時間が早いため、より多くの仕事ができるということを意味する。

ファインチューニングの方法

まずは最も一般的な**教師ありファインチューニング**（Supervised Fine-Tuning、SFTと略されます）を説明しましょう。

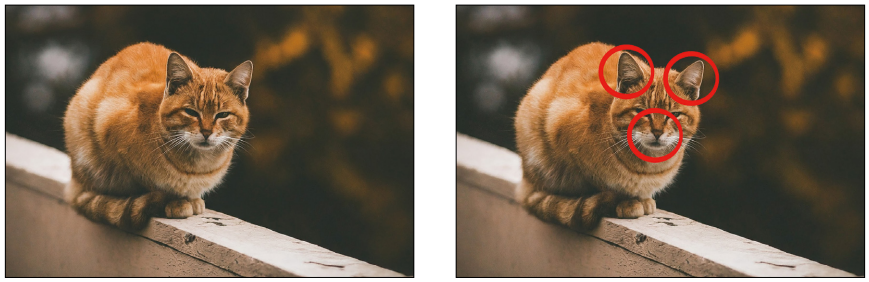
インストラクション・チューニングでは、大規模言語モデルを以下のようなテキストで追加学習します。データの与え方は異なりますが、学習自体は事前学習と同じで、テキストを生成する確率が高くなるように学習します。またファインチューニングではテキストの開始と終わりを正しく扱います。

38 注意機構（Attention）

人間の認知では目や耳からの情報すべてを使うのではなく、その一部分に注意するという脳の機構を認知科学や心理学では注意機構（Attention）と言います。この注意機構をモデル化することでニューラルネットワークの精度が大きく向上しました。

人間の認知と注意機構

■ 映像からネコを見出すときの注目点



例えば、この映像を見て猫が写っていると判断するとき、私たちはどの部分に注目しているでしょう。尖った耳と特徴的な目、丸い顔と鼻の形、ヒゲの生え方などから猫と判断できます。一方、背景や塀などは猫かどうかの判断に必要ない部分として無視されます。

こうした判断に必要な部分に注目し、それ以外を適切に無視するという認知の性質を**注意機構**（Attention）、あるいは単に注意と呼びます。雑踏や飲み会などまわりに騒音がある中で、会話相手の声を自然と聞き取れる人間の性質は注意機構の例によく挙げられます[1]。

[1] https://ja.wikipedia.org/wiki/カクテルパーティー効果

注意機構の基本

注意機構の考え方自体は以前からもありましたが、2010年代中盤からニューラルネットワークへの組み込みが活発になりました。

現在の大規模言語モデルの注意機構は複雑なので、まずは単一の注意機構からなる**Memory Network**を紹介しましょう。Memory Networkは、質問に対して適切な情報を記憶から引き出して回答するモデルです[2]。

Memory Networkが取り組む問題は、以下のような情報が与えられた状況で、"Where is Daniel?"（ダニエルはどこにいる？）という質問文に対して適切に回答するという質問応答タスクです。

1. Mary moved to the bathroom.（メアリーはバスルームに移動した）
2. John went to the hallway.（ジョンは廊下に行った）
3. Daniel went back to the hallway.（ダニエルは廊下に戻った）

この問題を次のようなステップで解きます。ます、この情報の中から質問文に関係がある情報を探します。次に同じDanielについて言及している情報3を見つけます。そして、その文章の情報から"hallway"（廊下）を答える、というステップです。

Memory Networkは、この解法手順をニューラルネットワークの計算に落とし込みます。Memory Networkでは、関連する情報をメモリー（記憶）に蓄え、ユーザからの質問に対してメモリーを検索しながら回答します。

[2] 自然言語処理の多くのタスクは質問応答の形に帰着できるという考え方が示されており、汎用人工知能（AGI）への一歩として提案されました。注意機構やPosition Encodingなど、現在も通用するさまざまな技術が採用されています。ここではMemory Networkの2つ目のバージョンであるEnd-to-End Memory Networkを紹介します。Sukhbaatar, Sainbayar, Jason Weston, and Rob Fergus. "End-to-End Memory Networks." Advances in neural information processing systems. 2015.

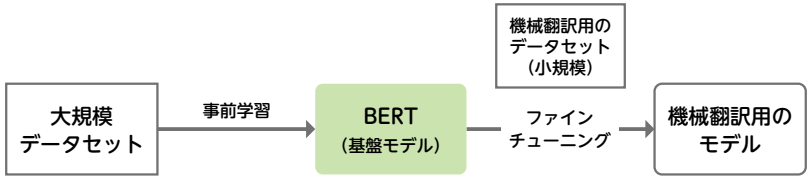
41 BERT

BERTは、後の大規模言語モデルの特性であるパラメータの多さと基盤モデル的振る舞いを備えたモデルです。BERTは自然言語処理にいくつものブレイクスルーをもたらしました。

BERT（バート）の特徴

BERT（Bidirectional Encoder Representations from Transformers、トランスフォーマーからの双方向エンコーダー表現）は、現在も続く基盤モデル（p.137参照）のアプローチを確固たるものにした画期的な言語モデルです[1]。BERTの衝撃は本当にすさまじく、当時の自然言語処理の論文のほとんどでBERTが使われていたと言っていいほどの勢いがありましたし、「BERT以前」「BERT以後」という言い方がよくされるほどでした。

■ BERTによる基盤モデルとファインチューニング



BERTは、トランスフォーマーのエンコーダー部分をそのまま利用したモデルです。つまりBERTの特徴はモデルの構造ではなく、事前学習（p.174参照）とファインチューニング（p.180参照）の学習フレームワークを確立した点にあります。

[1] Devlin, Jacob, et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805 (2018).

BERTの事前学習

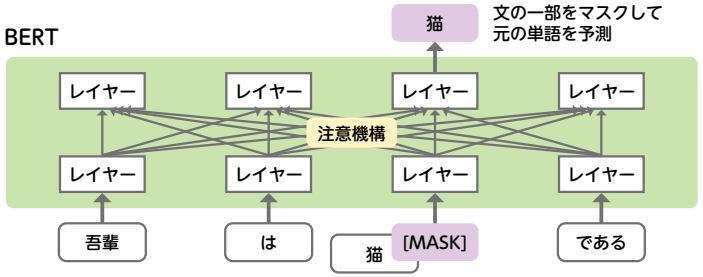
BERTの事前学習は、"Masked Language Model"と"Next Sentence Prediction"の2種類の自己教師あり学習（p.174参照）からなります。つまり、教師なしのデータから乱数などを使って生成した教師ありデータを用いて学習することで、大量のデータから低コストで精度の高い学習を行います。

Masked Language Model（マスクされた単語の予測）

Masked Language Model（マスクされた言語モデル）は単語の穴埋め問題を解かせる学習方法です。学校のテストで、文章の一部が空欄になっていて、適切な単語を選べという問題がよくありますよね。まさにそのイメージです。

テキストのいくつかの単語をランダムに選び、[MASK]という特別なトークンに置き換えてモデルに入力します。モデルは[MASK]の位置に入る単語の予測を出力します。これが元の単語になるように学習します。

■ BERTの事前学習（Masked Language Model）



図では、"吾輩は猫である"という文の"猫"をマスクしてBERTに入力しています。ここで、[MASK]の位置の出力が正解の"猫"になるようにモデルのパラメータを学習します。

この学習を大量のテキストに対して繰り返すことで、BERTは各単語がどのような文脈で使われるかを理解できるようになります。Word2Vec（p.119参照）も文脈（一緒に使われる単語）から学習していましたが、単語の順序は無視していました。BERTでは、語順も含めた文脈を元に学習します。

42 GPT (Generative Pre-trained Transformer)

OpenAIの大規模言語モデルであるGPTシリーズは、自然言語による指示でさまざまなタスクに対応できる画期的なモデルです。ChatGPTのエンジンとなり、生成AIブームを牽引しました。

GPTモデルの基本構造

GPT（Generative Pre-trained Transformer）は、トランスフォーマーのデコーダー部分を利用した、テキスト生成タスクに特化した自己回帰型の言語モデルです（p.145参照）。エンコーダー部分がないので、デコーダーからエンコーダーを参照する交差注意機構はなく、自己注意機構のみの構成になります。

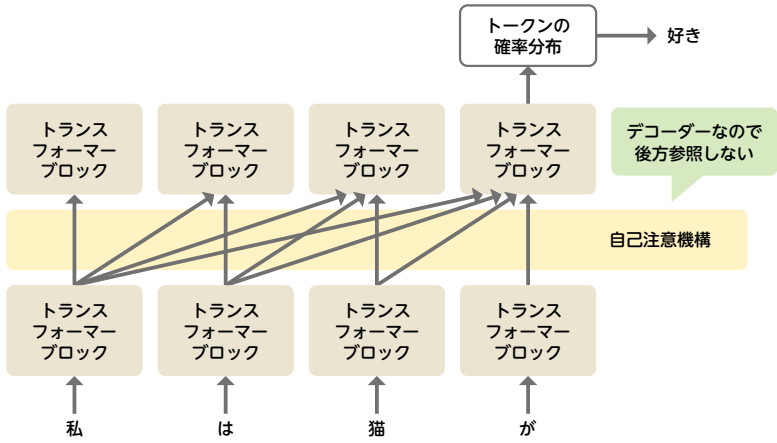
GPTには1から4までのバージョンがあります。

■ GPTのバージョン

GPT バージョン	発表年月	モデルサイズ
GPT	2018年6月	0.1B
GPT-2	2019年2月	1.5B
GPT-3	2020年6月	175B
GPT-3.5	2022年3月	350B
GPT-4	2023年3月	未公表（1.8T?）

GPTのバージョンが上がるたびに、そのモデルのサイズが増加しています。もともと深層学習のモデルは巨大でしたが、さらに桁違いに大規模なGPT-3は、自然言語の指示に従う能力を示し（創発性、p.141参照）、現在も続く生成AIブームの先駆けとなりました。そして後継のGPT-3.5はChatGPTの中核のエンジンとなりました。

■ GPT（Generative Pre-trained Transformer）の基本構成



基盤モデルとしての確立はBERTが先でしたが、その汎用性は追加学習（ファインチューニング）ありきのものでした。GPT-3はコンテキスト内学習（p.188参照）により自然言語でタスクの指示ができる汎用性を獲得しました。

GPT-3までのモデルはトランスフォーマーのデコーダーそのものであり、目新しい構造はありません。GPT-2とGPT-3の違いはほぼパラメータ数だけと考えられています。スケーリング則（p.140参照）の示すモデルサイズ・学習データ量・学習時間によってコンテキスト内学習が可能となったことを示したことは自然言語処理に衝撃を与えました。

GPT-3以降はモデルの詳細が徐々に公開されなくなり、GPT-4は公式にはモデルの構造に関する情報は一切公開されていません。リーク情報によれば、GPT-4は後述のMixture of Expertsを採用した1.8T（兆）パラメータのモデルだと言われています。

46 テキスト生成APIに指定するパラメータ

テキスト生成APIには、生成されるテキストの特性をコントロールするパラメータがあります。これらのパラメータを適切に設定することで、生成結果の質や一貫性を高めることができます。

テキスト生成APIのパラメータ

テキスト生成API（p.230参照）では、パラメータを設定することで生成テキストのバリエーションを増やしたり、逆にランダム性を抑えて、同じプロンプトに対して同じテキストを返すといった再現性を持たせたりできます。生成文をコントロールするという点で特に重要なパラメータは以下の3つです。

■ OpenAIテキスト生成APIの主なパラメータ

パラメータ	説明	値の範囲	デフォルト値
temperature	生成文のランダム度を指定します。高い値にすると多様な生成結果が得られます。	0.0 ～ 2.0	1.0
top_p	累積確率の閾値を設定します。1.0より小さくすると安定的なトークンが選ばれやすくなります。	0.0 ～ 1.0	1.0
max_tokens	生成されるトークン数の上限を設定します。	1以上の整数	（モデルの上限）

temperature（温度）

temperatureは「温度」の意味で、生成文のランダム度を指定するパラメータです。「温度」という用語は熱統計力学のモデルに由来します（p.148参照）。

temperatureが大きいと、スコアが小さいトークンにも確率が割り振られて、幅広いトークンが選ばれやすくなります。逆にtemperatureが小さいと、スコアが最大のトークンの確率は1に近づき、それ以外のトークンの確率は0に近づいて、トークンの確率の差が極端になります。特にtemperatureが0のときは、スコア最大のトークンが確率1となり、必ず選ばれます[1]。

temperatureを変えることで生成文にどのような影響があるか、実際の生成例で見てみましょう。temperatureを0から2の間で変えたとき、それぞれ3回の生成文を掲載しています[2]。

■「こんにちは、」で始まる文章を3回ずつ生成した冒頭

temperature	生成文
0.0	こんにちは、私はオープン AI の GPT-3 です。どのようにお ... こんにちは、私はオープン AI の GPT-3 です。どのようにお ... こんにちは、私はオープン AI の GPT-3 です。どのようにお ...
0.5	こんにちは、お元気ですか？　今日はどのようなご用件でし ... こんにちは、私は OpenAI の AI です。どのようにお手伝いでき ... こんにちは、私はオープン AI の言語モデルです。どのよ ...
1.0	こんにちは、お元気ですか？最近どんなことに取り組んで ... こんにちは、私は OpenAI の GPT-3 という AI モデルです。どの .. こんにちは、私はエミリーです。どのようにお手伝いで ...
1.5	こんにちは、私は人工知能の GPT-3 です。음 '> を '? 私は ... こんにちは、みなさん。 こんにちは、お元気ですか？私の名前はこれを ...
2.0	こんにちは、名称とう都备板，su 配置 -spacing-ind offshore... こんにちは、皆さん。才能 (さいだし グ -S.WriteByte(r)},Eff... こんにちは、{ :)California(Client_Ptr<G966/photos-by...

temperature=0.0では毎回同じ文章が生成されていますが、値が増えるごとに振れ幅が大きくなり、1.0より大きいとデタラメになっていくことがわかります。2.0では完全に壊れていますね。temperature単独で指定するなら1.0以下にするといいでしょう。

[1] temperatureは本来0にはなりませんが、temperature=0をスコア最大のトークンが決定的に選ばれる意味で用いる実装もあります。

[2] Completion APIでgpt-3.5-turbo-instructを使用。

52 AIの偏りとアライメント

統計的機械学習の目的は学習データから得られる分布を再現することなので、学習データに含まれる社会の偏りもまた再現してしまう可能性があります。そうしたAIの偏りがどういったリスクを生むか、どのような対策があるかを解説します。

学習データの偏りがAIに与える影響

大規模言語モデルは統計的機械学習（p.058参照）によって実現されています。その統計的機械学習の目的は、学習データから得られる分布を再現し、汎用的なモデルを獲得することです。この性質から問題となるのが、汎化（学習データに含まないデータの予測、p.064参照）と、学習データの**偏り**（bias）をそのまま継承してしまう点です。ここでは偏りの問題に注目しましょう。

まず学習データに含まれるヘイトや差別などの要素がモデルに反映される恐れがあります。具体的な事例として、MicrosoftのAIチャットボットTayが公開直後に停止されるという出来事がありました[1][2]。TayはMicrosoftが2016年に若者との対話を目的としてリリースしたチャットボットです。大量の公開データから学習されたTayは、ネオナチや反ユダヤ的なコメントを多数行うようになりました。MicrosoftはTayをオフラインにし、不適切な発言を行わないように調整しなければなりませんでした。

こうした明らかなヘイトや差別なら識別しやすいですが、それほど明確でない社会的な偏見もAIにとっては深刻な問題です。大規模言語モデルの学習データはインターネット上から集められたものが多いため、インターネット上で優勢な価値観をそのまま学習する可能性が高いです。以下はAIに反映される可能性のある社会のステレオタイプの例です。

[1] Tay (人工知能) - Wikipedia https://ja.wikipedia.org/wiki/Tay_(人工知能)

[2] Microsoftの人工知能Tay、悪い言葉を覚えて休眠中 - ITmedia NEWS https://www.itmedia.co.jp/news/articles/1603/25/news069.html

- **男性が科学技術や建設の仕事を、女性が看護や受付の仕事を担う**[3]
- **特定の民族や人種が特定の職業や役割に従事している**
- **若い人はテクノロジーに精通し、高齢者はそうでない**

これらは必ずしも差別ではないかもしれませんが、こうした偏りを持つAIは、少数派の意見が反映されない、少数の国や民族の文化が表現されなくなる、採用や融資などの判断において特定の人種や地域が不当に扱われる、などの公正でない結果をもたらす可能性があります。

現代社会がそうした偏見や差別を持っているのだから、AIが同じ価値観を持つのは仕方がない、という主張もありますし、ある程度は避けられない問題でもあるでしょう。しかしAIの利用が今後ますます拡大していく現状を考えると、この問題を放置せず、AIの価値観を是正する取り組みが求められます。

また生成AIは今後ますます普及し、インターネット上にはAIによって書かれたテキストが増えるでしょう。そうなると、AIの学習データにもAI由来のものが増え、AIの精度への悪影響が懸念されます[4]。さらに、学習データのフィードバックにより偏りを拡大再生産してしまう可能性も指摘されています[5]。

■ 偏りの拡大再生産

学習データから偏りを取り除くというシンプルな解決策は、そもそも偏りの検出が難しいことや、現在の大規模言語モデルでは精度が優先される事情から、現実的ではありません。大規模言語モデルの成功の要因の1つに、事前学

[3] Generative AI: UNESCO study reveals alarming evidence of regressive gender stereotypes | UNESCO https://www.unesco.org/en/articles/generative-ai-unesco-study-reveals-alarming-evidence-regressive-gender-stereotypes

[4] Alemohammad, Sina, et al. "Self-consuming generative models go mad." arXiv preprint arXiv:2307.01850 (2023).

[5] Taori, Rohan, and Tatsunori Hashimoto. "Data feedback loops: Model-driven amplification of dataset biases." International Conference on Machine Learning. PMLR, 2023.