

## ● 寄稿 1

# Pay Attention to 生成言語モデル ～セサミストリートを抜けるとそこには～

審査第二部 一般機械 前原 義明

### 抄録

生成言語モデルとは、入力された任意の文章から、当該任意の文章に続く文章を生成する、時に人工知能と呼ばれる言語モデルのことである。生成言語モデルとして最も有名なものはOpenAIが2020年に発表したGPT-3と呼ばれるモデルであるが、当該モデルのリリースを契機として、続々と他の生成言語モデルがリリースされ、各自然言語処理タスクにおけるモデルのパフォーマンスが信じられない速度で更新され続けている。本稿では、生成モデルの技術動向についてまとめるとともに、その代表的なユースケース、我々での活用可能性の検討等について紹介する。

## 1. はじめに

筆者は2020年7月から2022年6月までの2年間、Japio知財AI研究センター<sup>1)</sup>において、トランスフォーマーと呼ばれるニューラルネットを用いた、特許文書の機械翻訳をはじめとする特許・意匠・商標に関する人工知能の研究を行っていた。特に注力していた取組みは、特許ドメイン特化型BERTの開発及び当該BERT<sup>2)</sup>を用いたSDGs・脱炭素関連特許技術の解析である[Y. Maehara et al., 2022]が、これはトランスフォーマーのエンコーダー部分から派生したモデルを活用したものである。これに対し、トランスフォーマーのデコーダー部分から派生したGPT等に代表されるモデルは、俗に生成言語モデルと呼ばれているところ、GPT-3の発表以降、この技術が信じられない速度で発展してきている。筆者は、今こそ生成言語モデルに着目すべきであると考え、そのため、本稿においては、生成言語モデルの技術動向、代表的なユースケース等について紹介することとする。

## 2. 生成言語モデルとは

生成言語モデルとは、OpenAIのGPTシリーズに代表される、ある任意の文章からその文章に続く文字列を予測する言語モデルのことをいう。

この文字列の生成は、ある文章の次にくる単語(トークン)を予測する処理の連続によって行われるものである。この処理は、具体的には、モデルが持つ辞書データのそれぞれの単語が、次に出力される単語に該当する確率を算出することによって実現される。GPTシリーズにおいては、そのモデルアーキテクチャがトランスフォーマーデコーダーであるため、トランスフォーマーブロックの出力Logitsをsoftmax関数で正規化することによって、この確率が算出されている。

## 3. GPTシリーズ

生成言語モデルで最も有名なものはOpenAIが発表したGPT[A. Radford et al., 2018]である。

1) <https://transtool.japio.or.jp/work/>

2) Bidirectional Encoder Representation for Transformersの略称であり、セサミストリートのパートから命名された。

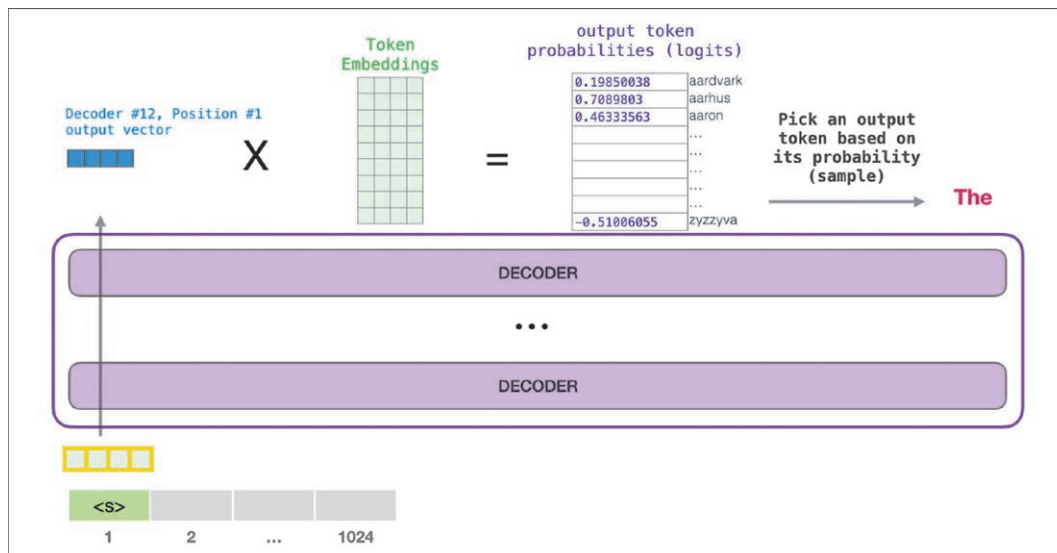


図1 GPTモデルの文章生成概略図 (出典：The Illustrated GPT-2<sup>3)</sup>)

GPTとは、Generative Pre-trained Transformerの略称であり、モデルに入力された任意の文章から、当該文章の次に続く文字列を予測する機能を持っている。2019年に発表されたGPTの後続モデルであるGPT-2 [A. Radford et al., 2019]については、フェイクニュース拡散の懸念から、OpenAIがそのオープンソース化を拒んでいたが、最終的にはモデルもその訓練コードも公開されるに至っている<sup>4)</sup>。また、訓練コードがオープンソース化されたため、オリジナルの英語モデルだけでなく、日本語を含む様々なモデルが公開されている。日本語GPTモデルで最も有名なものは、rinna社(旧Microsoft社)が開発したものである<sup>5)</sup>。当該モデルは、以下の通り、流暢な日本語の文章を生成することができる。(緑マーカー部がrinna社の日本語GPT-2による出力)

rinna社の日本語GPT-2による出力

生命、宇宙、そして万物についての究極の疑問の答えは、わたしたちも知っている。わたしたちは自然と、人類の歴史と科学から生まれた知恵である。われわれはすでに、すべての生命を起源とする生命をどのようにして創った

#### 4. GPT-3

生成言語モデルを取り巻く現状が変わったのは、2020年におけるGPT-3 [T. B. Brown et al., 2020]のリリースからである。GPT-3の特徴は、従来のGPTモデルを大きくスケールアップした点にある。BERTやGPTに代表されるTransformerモデルは、Full Attentionと呼ばれるアーキテクチャを採用しているため、モデルサイズがトークン長の二乗に比例し、スケールアップが非常に困難であったが、OpenAIはGPTにSparse Transformer [R. Child et al., 2019]というアーキテクチャを採用することによってこの問題を解決した。図2はSparse TransformerにおけるAttention機構の概略図である。Sparse Transformerでは、ある特定の位置に係るアテンションマップを定義する際、直前の数個分及び数個おきのチェックポイントにしか着目しない(図2においては直前3個の位置及びそれらより前の4個おきの位置)。TransformerやBERTで用いられているFull Attentionというメカニズムは、入力データ同士の全ての関係を捉えるため、パラメータサイズが $O(n^2)$ ( $n$ は入力データ長)となっていたが、Sparse TransformerのAttention機構であれば、 $O(n)$ で済む。これにより、従来よりも大きなス

3) <https://jalamar.github.io/illustrated-gpt2/>

4) <https://github.com/openai/gpt-2>

5) <https://huggingface.co/rinna/japanese-gpt2-medium>

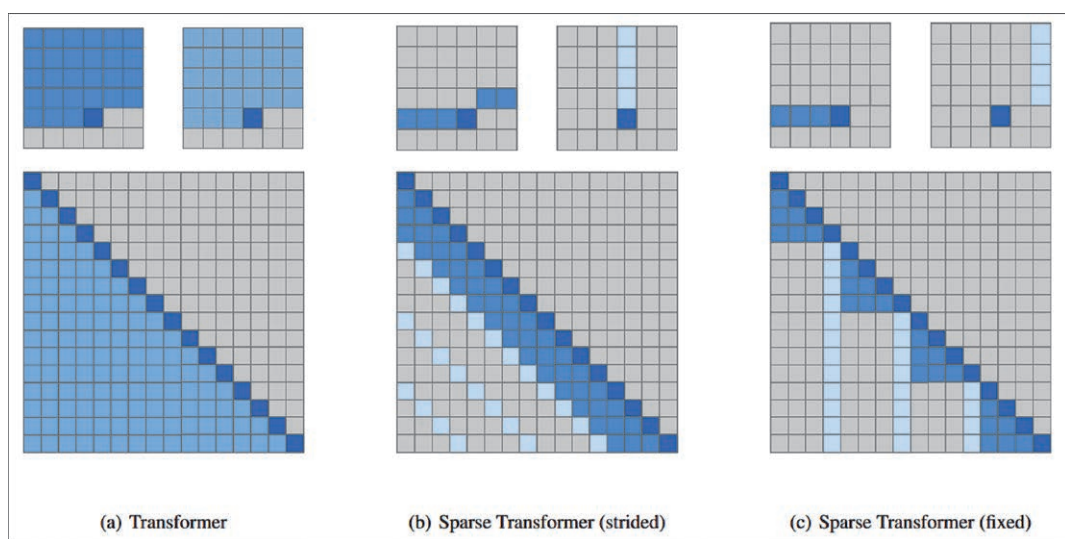


図2 Sparse TransformerにおけるAttention機構

スケールアップが可能となった。なお、同様のAttention機構を導入したBERTの亜種（BigBird）もリリースされており、要約タスク等の長い入力データの処理が求められるようなタスクにおいて、従来のモデルを大きく超えるような精度がこれらのモデルによって叩き出されているところである [M. Zarheer et al., 2020]。

GPT-3の論文である「Language Models Are Few-shot Learners」の内容は、近年における他のエポックメイキングな論文（Transformer、BERT等）とは異なり、GPTモデルをスケールアップした点及び当該スケールアップした結果、どのような成果があったのかという点について淡々と紹介する内容になっている。結果として、GPT-3は、768トークン長・48層・15億パラメータであったGPT-2に対して、2,048トークン長・96層・1,750億パラメータまでスケールアップされている。

### 5. Fine-tuning vs. Few-shot/One-shot/Zero-shot

GPT-3の論文では、従来のFine-tuningというモデルの学習方法とは全く異なる、Few-shot/One-shot/Zero-shotと呼ばれる学習手法が提案されている。Fine-tuningとは、BERTを用いた自然言語処理タスクへのアプローチに代表されるような、訓練推論を行う度に損失をモデルに逆伝播させてモデルパラメータの更新を行う学習方法である。それに対

し、OpenAIが当該論文にて提唱したFew-shot/One-shot/Zero-shotと呼ばれる学習方法は、訓練推論の後にモデルパラメータの更新を行わない。ではどうしているのかということ、Few-shotでは、以下に示す図のように、いくつかのタスクの例示及び質問文をプロンプトとしてGPT-3に与えるのである。スケールアップされたモデルであれば、与えられたタスクの例示を参照しながら、最後の質問に対する解答を生成することができる。また、One-shotはタスクの例示を1つしか行わないもの、Zero-shotはタスクの例示を行わずにタスクの説明だけで推論を行わせるものである。以下は論文で例示されている各学習例をOpenAI APIで再現したものである。英単語を仏単語に変換する等の比較的簡単なタスクであれば、難なくこなせていることが理解できる。

#### Few-shot (英語から仏語へ)

```
Translate English to French:
sea otter => loutre de mer
peppermint => menthe poivrée
plush giraffe => giraffe peluche
cheese => fromage
```

#### One-shot (英語から仏語へ)

```
Translate English to French:
sea otter => loutre de mer
cheese => fromage
```

## Zero-shot (英語から仏語へ)

```
Translate English to French:  
cheese => fromage
```

なお、論文中における英仏単語翻訳の例示を英日単語翻訳に変更しても、以下の通り、適切に翻訳が行われることを確認できた。

## Few-shot (英語から日本語へ)

```
Translate English to Japanese:  
sea otter => ラッコ  
peppermint => ペパーミント  
plush giraffe => キリンのぬいぐるみ  
cheese => チーズ
```

## One-shot (英語から日本語へ)

```
Translate English to Japanese:  
sea otter => ラッコ  
cheese => チーズ
```

## Zero-shot (英語から日本語へ)

```
Translate English to Japanese:  
cheese => チーズ
```

表1に示す通り、GPT-3が学習したコーパスに日本語はほとんど含まれていないため(約0.1%)、この挙動は驚くべきものである。

表1 GPT-3のコーパスにおける各言語の単語単位の含有率(出典: OpenAI)

言語	単語総数	単語総数割合
en (英語)	181014683608	92.64708%
fr (フランス語)	3553061536	1.81853%
de (ドイツ語)	2870869396	1.46937%
ja (日本語)	217047918	0.11109%

## 6. OpenAI API

GPT-3はOpenAI API<sup>6)</sup>にアカウントを作成すれば、誰でも利用することができる。GPT-3のリリース直後は申請後のwaitlistの通過を経ないとユーザーとなることができなかったが、技術のコモディティ化に伴い、現在ではwaitlistは廃止されている。そのため、申請後の即利用が可能である。利用はPythonのopenaiというパッケージ経由でターミナル上からAPIを呼び出すか、PlayGroundと呼ばれるWebAPI上にプロンプトを入力することによって行う。

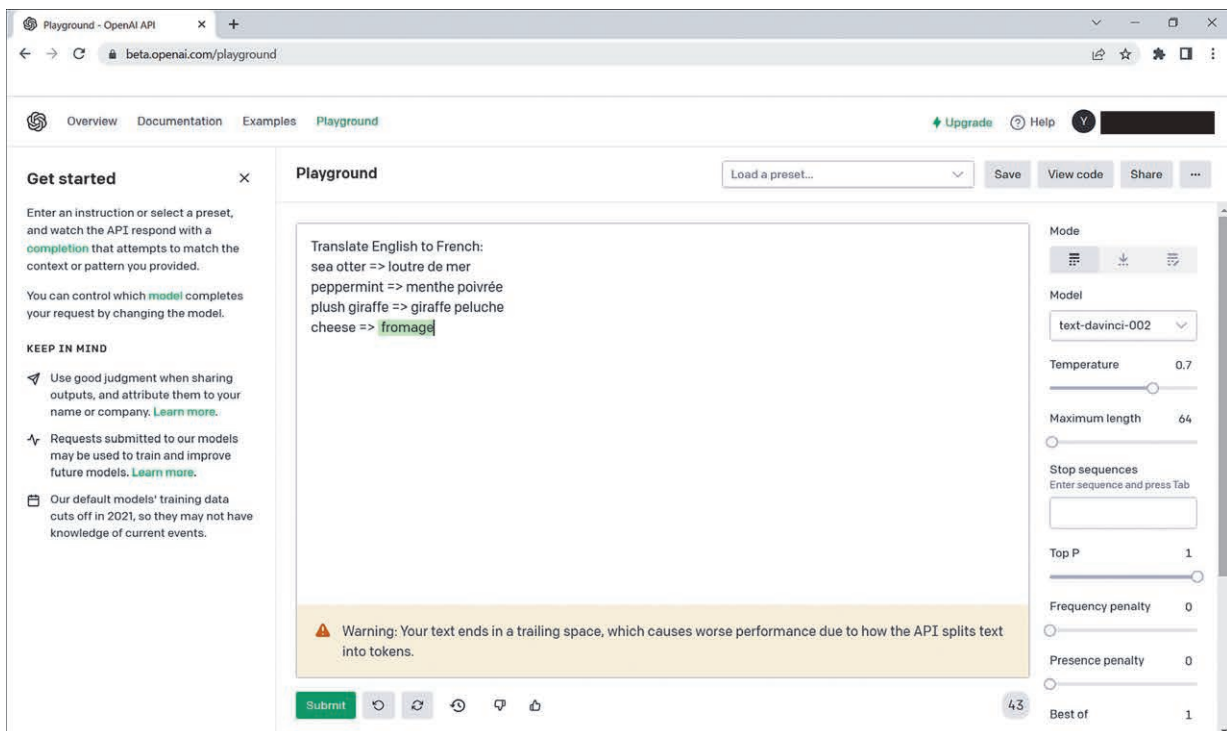


図3 OpenAI APIにおけるPrayground

6) <https://openai.com/api/>

利用に対する課金は従量課金制であり、利用料は以下の通りである(2022年9月時点)。

表2 OpenAI API Pricing (出典: OpenAI)

Ada	Babbage	Curie	Davinci
\$0.0004/1K tokens	\$0.0005/1K tokens	\$0.0020/1K tokens	\$0.0200/1K tokens

OpenAI APIで利用可能な各モデルには、過去の科学者の名前が付けられている。最も高額なモデルはDavinci (レオナルド・ダ・ヴィンチ) と呼ばれる1,750億パラメータからなるモデルであり、Babbage (チャールズ・バベッジ)、Curie(マリー・キュリー)、Ada(エイダ・ラブレス)と続いてゆく。上述の通り、コーパスにおける日本語の含有率が非常に低いため、日本語のタスクをこなさせるには、最上位モデルのDavinciを利用することが望ましい。利用料は2022年9月に値下げがなされており、技術の進展に伴い、今後も利用料は下がっていくことが予想される。GPT-3にタスクを処理させる場合は、プログラミングではなく、プロンプトを設計することによって行われるが、OpenAI APIにはいくつかのプロンプト例(チャットボット、SQL文作成等)が用意されており、プロンプトの設計に際してはこれらを参考にすればよい。以下は、チャットボットのプロンプト例にならって作成した、旧特技懇誌における「ギコンくんがいく」である。

GPT-3による「ギコンくんがいく」

ギコンくん：コレじいさん、質問です！この前の回で災害対策の話になってましたけど、災害で停電になっちゃっても特許庁は大丈夫なのですか？

コレじいさん：むむむ、いいところに気が付いたな、ギコンくん。電子システムが組織の心臓になっている特許庁では、停電は大問題じゃ！そんなときのために、ガスタービン発電からなる非常用発電設備を備えておるのじゃよ。

ギコンくん：パテッ！そんなものがあるんですか？！初耳です！どこにあるんですか？

コレじいさん：聞いておくんだな、ギコンくん。それは、特許庁の裏手にある屋上にあるの

じゃ。

ギコンくん：屋上に???

コレじいさん：そうじゃ。実は、特許庁は東京都千代田区の一筆書きで建物が建てられておるのじゃ。だから、屋上には何も無いのじゃ。

ギコンくん：そうなんですか…

実際には、特許庁の非常用発電設備は特許庁の地下に存在するものの、一応、ギコンくんとコレじいさんのそれっぽい会話を生成することができた。

また、論文中のFew-shotの要領で特許分類(IPCサブクラス)の推定も可能であった。これは、ウェブクロールされたGPT-3のコーパスの中にIPCサブクラスに関する日本語特許公報の情報が含まれていたためか、大量に含まれている英語の情報から類推しているか、であると考えられる。

GPT-3による特許分類推定(IPCサブクラス)

特許分類
自動運転 => G08G
ロケット => B64G
洗濯機 => D06F

また、2021年より、API経由でのモデルのFine-tuningの機能も提供されている。その利用は、APIキーをローカルの環境変数として保存して、Pythonのopenaiパッケージ<sup>7)</sup>を用いることによって可能となる。Fine-tuningはGPT-3を下流タスクに適合させる上で非常に有用なオプションではあるが、以下の通り金額コストがかかるため、実施の際には注意が必要である。また、学習パラメータについても細かく指定することはできず、学習率及びエポック数が固定されてしまっているのが難点である。

表3 OpenAI APIにおける各GPT-3モデルのFine-tuningコスト(出典: OpenAI API Pricing<sup>8)</sup>)

モデル	訓練コスト	利用コスト
Ada	\$0.0004/1K tokens	\$0.0016/1K tokens
Babbage	\$0.0006/1K tokens	\$0.0024/1K tokens
Curie	\$0.0030/1K tokens	\$0.0120/1K tokens
Davinci	\$0.0300/1K tokens	\$0.1200/1K tokens

7) <https://github.com/openai/openai-python>  
 8) <https://openai.com/api/pricing/>

## 7. トランスフォーマーのスケールング則

GPT-3の性能を見せつけられると、生成言語モデルのパフォーマンスが今後どこまで上がるのかという点について興味が出てくるだろう。この点については、OpenAIがGPT-3の論文と同時期に発表した論文 [J. Kaplan et al., 2020] によって言及されている。結論からいえば、GPT-3に代表される言語モデルの性能には限界がない。OpenAIの当該論文では、Transformerのスケールング則という現象について言及されている。スケールング則とは、「Transformerの性能がパラメータ数・データサイズ・計算資源のたった3つの変数によって支配されている。」というものである。図4に示す通り、水平軸にそれぞれ3つのパラメータのべき乗をとってモデルの損失をプロットすると、綺麗な直線になっていることが理解できる。

これはつまり、これら3つのパラメータを大きくし続けていけば、Transformerの性能が際限なく上が

り続けていくことを意味している。スケールング則の発見によって、生成言語モデルに代表される人工知能の開発における技術課題は、「どれだけの投資ができるか？」という政治的な問題となった。そのため、スケールング則に関するOpenAIの論文の発表から、各企業が発表したモデルのパラメータは加速度的に増加し続けている。さらに驚くべき点は、言語だけでなく、画像や音声等のTransformerが適用される全てのモダリティにおいてスケールング則の適用があることである。今後は自然言語だけでなく、全てのモダリティにおいて、GPT-3の発表時のような衝撃が待っている可能性がある。

## 8. GPT-3 亜種

GPT-3及びスケールング則の発表を受けて、GoogleやDeepMindをはじめとする他の企業も、続々とGPT-3亜種と呼びうる独自の生成言語モデルを発表した。以下、それらについて紹介する。

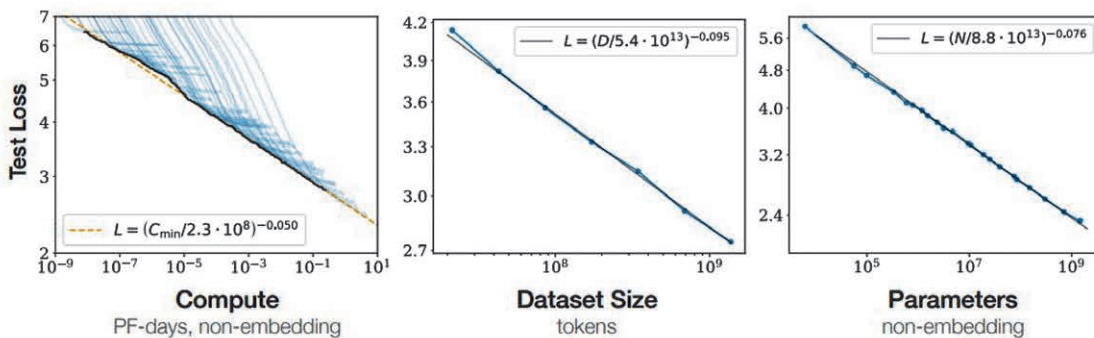


図4 計算資源・データサイズ・パラメータ数に対する損失の変化

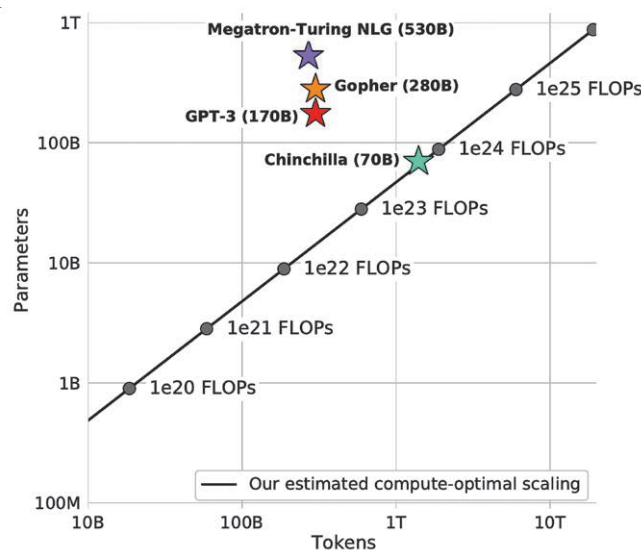


図5 各種GPT-3亜種のコーパストークン数及びパラメータ数 (出典: DeepMind)

## 8.1 GPT-J及びGPT-NeoX

EleutherAIという研究者団体によって訓練された、GPT-3と同様のモデルアーキテクチャを採用した生成言語モデル。GPT-Jのパラメータ数は60億、GPT-NeoXは200億であり、GPT-3に比べるとパフォーマンスの点で見劣りするが、そのため逆に、TeslaA100のような汎用GPU単騎でのFine-tuningが可能である。GPT-Neoについては、日本語版もリリースされている<sup>9)</sup>。

## 8.2 Gopher [J. W. Rae et al., 2022]

AlphaGoを開発したイギリスのDeepMind社によって発表された生成言語モデル。Gopherの学習において、モデルパラメータ数は2,800億までスケールアップされており、GPT-3等で報告されていた生成言語モデルのバイアスを抑制するために、有害なコーパスをクリーニングする等の工夫がなされている。Gopherは、152にも及ぶ様々なタスクにおいてSoTA (State of The Art) を達成している。論文においては、文章理解やファクトチェック等のタスクにおいては、モデルのスケールアップが有意に働くが、数学の文章問題に代表されるような推論型のタスクでは、モデルのスケールアップが精度向上にあまり寄与しなかったことが報告されている。

## 8.3 Chinchilla [J. Hoffmann et al., 2022]

DeepMindによってトレーニングされた生成言語モデル。パラメータ数は700億であり、GPT-3やGopherに比べて劣るものの、Gopherに対して4倍以上の訓練データスケールが行われており、様々な下流タスクにおいてGopherを含む既存のモデルに対する優位性が示されている。

## 8.4 HyperCLOVA

LINE株式会社によって開発された日本語コーパスを中心にトレーニングが行われた日本語生成言語モデル。モデルはNVIDIAのMegatron-LMであり、

パラメータ数は390億である。現在、一般向けの公開はされていない。

## 8.5 PaLM [A. Chowdhery et al., 2022]

6,144ものTPU<sup>10)</sup>を用いてGoogleによって学習された、5,400億パラメータからなる現時点における最大規模の生成言語モデル。PaLMにおいては、chain-of-thought prompting [Wei et al., 2022] という推論手法が用いられている<sup>11)</sup>。これは、クエリの質問文から回答を直接生成するのではなく、思考過程を表現する文章をまず生成してから、回答を生成する手法である。PaLMのような超巨大生成言語モデルでchain-of-thought promptingを用いることで、算数の文章問題のような回答を行うに際して思考の連鎖が要求されるようなタスクにおいて、目覚ましい成果が出たことが報告されているところである。

## 8.6 BLOOM

Transformers<sup>12)</sup>で有名なHuggingFaceによるオープンソース化された生成言語モデル。パラメータ数は最大のもので1,760億。BLOOMの利用は、HuggingFaceへのユーザ登録をすることによって可能となるが、OpenAI APIと異なり無料で利用することが可能である。BLOOMの事前学習は、384個のGPU(A100)を3.5カ月も用いて行われているが、当該事前学習の経緯についてまとめた研究者のブログ<sup>13)</sup>が示唆に富んでおり非常に参考になる。

## 9. 生成言語モデルのユースケース

GPT-3に代表される生成言語モデルは、BERTに代表されるベクトルが出力となるエンコーダーモデルとは異なり、文章が出力されるので出力の多様性が担保されている。そのため、生成言語モデル自体の進化と同様に、生成言語モデルのユースケースも近年大きな進化を見せている。以下、代表的なものを紹介する。

9) <https://huggingface.co/yellowback/gpt-neo-japanese-1.3B>

10) Tensor Processing Unit

11) <https://ai.googleblog.com/2022/05/language-models-perform-reasoning-via.html>

12) <https://huggingface.co/docs/transformers/index>

13) <https://huggingface.co/blog/bloom-megatron-deepspeed>

## 9.1 機械翻訳モデルの蒸留 [J. M. Han et al., 2021]

通常、機械翻訳モデルの訓練は、研究者によって構築された対応する対象二言語対の集合、いわゆる対訳コーパスを用いて行われるが、当該論文ではGPT-3から知識蒸留を行うことによって機械翻訳モデルを作成している。作成の方法は、まず、以下のZero-shotプロンプトでGPT-3からフランス語と英語の対訳文対を出力させる。

```
<sep> Given the following passage in  
<srcLang>: <sep> <srcSeq> <sep> a good  
<tgtLang> translation is: <sep> <tgtSeq>  
<sep>.
```

次は、このZero-shotプロンプトによって生成されたいくつかの対訳文対から、図6のようなFew-shotプロンプトを作成することによって対訳コーパスをGPT-2によって生成させる。

最後は、この対訳コーパスを使ってGPT-3をFine-tuningすることによって機械翻訳モデルをトレーニングする。この機械翻訳モデルを作成するフローでは、対訳コーパスを作成するための人的な翻訳処理は一度も出現せず、GPT-3が行う文章生成処理によって全てが完結しているが、当該論文は、この手法により、WMT14の英仏翻訳タスクにおけるBLEU値42.1を達成している。

## 9.2 AlphaCode [Y. Li et al., 2022]

生成言語モデルはコンピュータプログラムのコードも学習しているため、C言語やPythonといったプログラミング言語のコードを生成することもできるが、OpenAIは競技プログラミングの問題を生成言語モデルに解かせることに成功した。これはGitHubの存在する大量のコードを生成モデルに事前学習させ、Codeforces<sup>14)</sup>等にある競技プログラミングの問題文及び回答例のコードで当該モデルをFine-tuningするというものである。以下に競技プログラミングの問題例とAlphaCodeが生成したプログラムを示す。

### Codeforcesの問題例

(出典：Competitive programming with AlphaCode<sup>15)</sup>)

#### Backspace

You are given two strings  $s$  and  $t$ , consisting of lowercase English letters. You are going to type the string  $s$  character, from first character to the last one.

When typing a character, instead of pressing the button corresponding to it, you can press the “Backspace” button. It deletes the last character you have typed among those that aren’t deleted yet (or does nothing if there are no characters in

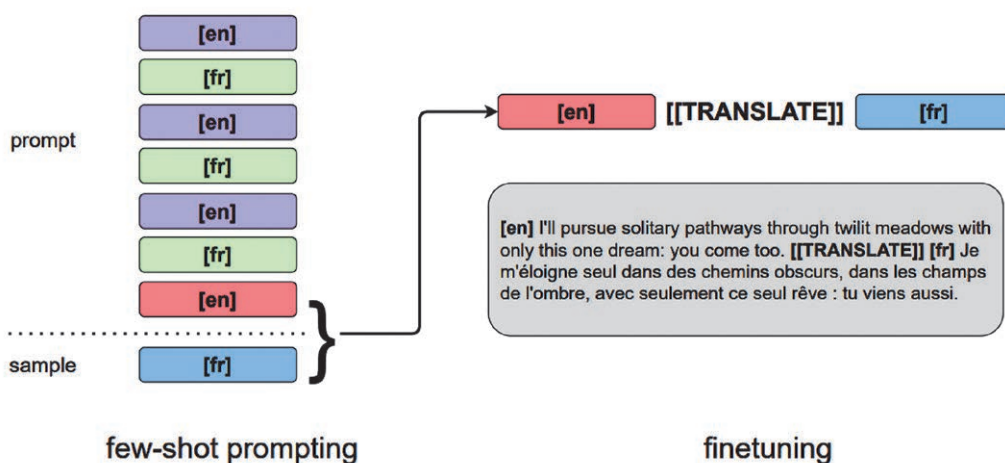


図6 Few-shot promptingによる対訳コーパス生成 (出典：OpenAI)

14) <https://codeforces.com/>

15) <https://www.deepmind.com/blog/competitive-programming-with-alpha-code>



the current string.). For example, if *s* is “abcd” and you press Backspace instead of typing the first and the fourth characters, you will get the string “bd” (the first press of Backspace deletes no character ‘c’). Another example, if *s* is “abcaa” and you press Backspace instead of the last two letters, then the resulting text is “a”.

Your task is to determine whether you can obtain the string *t*, if you type the string *s* and press “Backspace” instead of typing several (maybe zero) characters of *s*.

**Input**

The first line contains a single integer  $q(1 \leq q \leq 10^5)$  the number of test cases. The first line of each test case contains the string  $s(1 \leq |s| \leq 10^5)$ . Each character of *s* is a lowercase English letter.

The second line of each test case contains the string  $t(1 \leq |t| \leq 10^5)$ . Each character of *t* is a lowercase English letter.

It is guaranteed that the total number of characters in the strings over all test cases does not exceed  $2 \cdot 10^5$ .

AlphaCodeの回答例

(出典 : Competitive programming with AlphaCode)

```
t=int(input())
for i in range(t):
    s=input()
    t=input()
    a=[]
    b=[]
    for j in s:
        a.append(j)
    for j in t:
        b.append(j)
    a.reverse()
    b.reverse()
```

```
c=[]
while len(b)!=0 and len(a)!=0:
    if a[0]!=b[0]:
        c.append(b.pop(0))
        a.pop(0)
    elif a[0]!=b[0] and len(a)==1:
        a.pop(0)
        a.pop(0)
    elif a[0]!=b[0] and len(a)==1:
        a.pop(0)
if len(b)==0:
    print("YES")
else:
    print("NO")
```

なお、競技プログラマ（競プロer）としての能力はレートで表現されることができているが、現時点でのAlphaCodeレートは約1,300（基礎的なアルゴリズム処理能力については疑いのないレベル）まで到達している。

**9.3 WebGPT [R. Nakano et al, 2021]**

GPT-3にブラウザを操作させるコマンドを生成することによって、任意の質問文に対してウェブから得られる情報を用いて回答できるようにしたGPT-3のユースケースである。WebGPTはプロンプトに含まれる、質問、今まで引用した文章、アクション履歴、ウェブページのタイトル、スクロールバーの位置、ブラウザが表示しているテキスト、残りアクション回数、次のアクションという項目から以下の表に示されるコマンドを生成する。（これら以外のコマンドを出力した場合、操作は無効とされる。）

WebGPTはオープンソース化されていないが、WebGPTが作成した任意の質問文への回答についてはいくつかのサンプルが公開されている<sup>16)</sup>。次項は、「How do neural networks work?」（ニューラルネットワークはどうやって動くの?）というオープンクエスチョン型の質問に対するWebGPTが作成した回答である。チェリーピックではあろうものの、非常に論理的な回答が生成されていることが理解できる。

## WebGPTがプロンプトから生成するコマンド

コマンド	内容
Search <query>	BingAPIにサーチクエリ<query>を送り、検索結果を表示する。
Clicked on link <link ID>	リンクを辿って新しいページに行く。
Find in page: <text>	<text>を現在のページの中で検索し、そこまでスクロールする。
Quote: <text>	もし、<text>が現在のページで発見されれば、それを引用リストに加える。
Scrolled down <1, 2, 3>	番号の数だけ下スクロールする。
Scrolled up <1, 2, 3>	番号の数だけ上スクロールする。
Top	ページトップに行く。
Back	前のページに行く。
End: Answer	ブラウジングを終了し、回答フェーズに移行する。
End: <Nonsense, Controversial>	ブラウジングを終了し、回答フェーズをスキップする。

## WebGPTの解答例

In its most basic form, a neural network has two layers: an input layer and an output layer [3]. The output layer is the component of the neural network that makes predictions [3]. In a feedforward network, information flows through the network in the following way: patterns of information are fed into the network via the input units, which trigger the layers of hidden units, and these in turn arrive at the output units [1]. The network learns by a feedback process called backpropagation, which involves comparing the output a network produces with the output it was meant to produce, and using the difference between them to modify the weights of the connections between the units in the network, working from the output units through the hidden units to the input units, going backward [2] [4]. Over time, backpropagation causes the network to learn, reducing the difference between actual and intended output to the point where the two exactly coincide, so the

network figures things out exactly as it should [2].

1. How neural networks work - A simple introduction (www.explainthatstuff.com)
2. How neural networks work - A simple introduction (www.explainthatstuff.com)
3. How Do Neural Networks Really Work? | Nick McCullum (nickmccullum.com)
4. How Do Neural Networks Really Work? | Nick McCullum (nickmccullum.com)

なお、WebGPTのアイデアは特許審査や商標の審査にも適用可能であると筆者は考えている。

### 9.4 Minerva [A. Lewkowycz et al., 2022]

上述のPaLMをベースに、chain-of-thought promptingというプロンプトエンジニアリングによって、数学の問題を解けるようにした生成言語モデルである。PaLMから進歩している点は、科学技術論文等のコーパスを読み込んでいる点にある。Minervaの回答例は、Google ResearchのGitHubリポジトリ<sup>17)</sup>で参照することが可能である。一例を図7に示す。

米国高校レベルの数学問題を集めたMATHという

16) <https://openai.com/blog/webgpt/>

17) <https://minerva-demo.github.io/>

**Question:** A particle moves so that it is at  $(3 \sin \frac{t}{4}, 3 \cos \frac{t}{4})$  at time  $t$ . Find the speed of the particle, measured in unit of distance per unit of time.

---

**Model output:** The speed is the magnitude of the velocity vector. Therefore, we must first find the velocity vector, which is the derivative of the position vector. We have

$$\mathbf{v} = \frac{d}{dt} \left( 3 \sin \frac{t}{4}, 3 \cos \frac{t}{4} \right) = 3 \left( \cos \frac{t}{4}, -\sin \frac{t}{4} \right) \frac{1}{4} = \left( \frac{3}{4} \cos \frac{t}{4}, -\frac{3}{4} \sin \frac{t}{4} \right).$$

Therefore, the magnitude of the velocity vector is

$$|\mathbf{v}| = \sqrt{\left( \frac{3}{4} \cos \frac{t}{4} \right)^2 + \left( -\frac{3}{4} \sin \frac{t}{4} \right)^2} = \sqrt{\frac{9}{16} \cos^2 \frac{t}{4} + \frac{9}{16} \sin^2 \frac{t}{4}}$$

$$= \frac{3}{4} \sqrt{\cos^2 \frac{t}{4} + \sin^2 \frac{t}{4}} = \frac{3}{4}.$$

Therefore, the speed of the particle is  $\frac{3}{4}$ .

図7 Minervaによる数学問題への解答例 (出典: Minerva Explorer)

データセットにおいて、Minervaは50.3%という正答率を叩き出している。この結果をみると、「ついにここまで来たのか。」と思わずにいられない。

### 10. 生成言語モデルのトレーニング

最後に生成言語モデルを実際にトレーニング(事前学習)する方法を紹介したいと思う。個人でパラメータ数が億を超えるような大規模モデルをトレーニングする際は、基本的にはAWS<sup>18)</sup>でTeslaV100等の高性能GPUを時間借りすることによって行う。(そのような場合でも、予算の関係から損失が収束するようなどころまで学習を行うことは不可能である。)ここでは、HuggingFaceのコードでrinna社がリリースしている13億パラメータの日本語GPTを事前学習する方法を紹介したいと思う。rinna社のGPTモデルはHuggingFaceが提供するTransformersというライブラリで利用・訓練することが可能である。HuggingFaceのコードは以下の通り、GitHub<sup>19)</sup>からダウンロードすることができる。

```
$ pip install transformers datasets evaluate
$ git clone https://github.com/huggingface/transformers
$ cd transformers/examples/pytorch/language-modeling
```

トークナイザ及びコーパスにrinna社のGPTモデルと同じものを流用した場合、HuggingFaceの生成言語モデル用トレーニングコードを実行するためのスクリプトは以下の通りとなる。(最低限のオプションのみ)なお、CC100のデータセットの日本語Verを読み込むように、Python実行コード内のデータ読み込みクラスの引数に日本語フラグ(ja)を設定する必要がある。

```
python run_clm.py \
  --model_name_or_path rinna/japanese-gpt-1b \
  --tokenizer_name rinna/japanese-gpt-1b \
  --dataset_name cc100 \
  --per_device_train_batch_size 1 \
  --per_device_eval_batch_size 1 \
  --do_train \
  --do_eval \
  --output_dir /tmp/test-clm
```

前処理に非常に時間がかかってしまい、途中でp2インスタンスの利用を打ち切ってしまったため、残念ながらこの訓練の結果を本稿で紹介することはできない。しかし、基本的にはこれでrinna社と同じスペックのGPTモデルをトレーニングすることが可能となる。

18) Amazon Web Service  
 19) <https://github.com/huggingface/transformers>

## 11. まとめ

本稿では、生成言語モデルの概要、代表的なユースケース等について紹介した。紹介の通り、Transformerのエンコーダーモデルの応用であるBERT等と比べて、生成言語モデルはその出力が任意の文章であるため、非常に広範な分野へ応用が可能である。また、その性能についても、上述のスケールリング則によって、予算を投入し続ければ上がり続けることが示されており、今後、人類のいわゆる「知的」であると言われていた作業が、生成言語モデルによって代替されていくことが予想される。今後も一特許審査官として、この技術の発展に注意を向け続けていきたい。

(本稿における見解は著者個人のものであり、著者が所属する組織のものではない。)

### 参考文献

- [1] Y. Maehara, A. Kuku, Y. Osabe, Macro analysis of decarbonization-related patent technologies by patent domain-specific BERT, 69, World Patent Information, 2022, 102112, <https://doi.org/10.1016/j.wpi.2022.102112>.
- [2] A. Radford et al., Improving Language Understanding by Generative Pre-Training, 2018, [https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language\\_understanding\\_paper.pdf](https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf).
- [3] A. Radford et al., Language Models are Unsupervised Multitask Learners, 2019
- [4] T. B. Brown et al., Language Models are Few-Shot Learners, 2022, [https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language\\_understanding\\_paper.pdf](https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf).
- [5] R. Child et al., Generating Long Sequences with Sparse Transformers, 2019, <https://arxiv.org/abs/1904.10509>
- [6] M. Zaheer et al., Big Bird: Transformers for Longer Sequences, CoRR, 2020 [abs/2007.14062](https://arxiv.org/abs/2007.14062), <https://arxiv.org/abs/2007.14062>.
- [7] J. Kaplan et al., Scaling Law for Neural Language Models, 2020, <https://arxiv.org/abs/2001.08361>.
- [8] J. W. Rae et al., Scaling Language Models: Methods, Analysis & Insights from Training Gopher, 2022, <https://arxiv.org/abs/2112.11446>.
- [9] J. Hoffmann et al., J. W. Rae et al., Training Compute-Optimal Large Language Models, 2022, <https://arxiv.org/abs/2203.15556>.
- [10] A. Chowdhery et al., PaLM: Scaling Language Modeling with Pathways, 2022, <https://arxiv.org/abs/2204.02311>.
- [12] J. M. Han et al., Unsupervised Neural Machine Translation with Generative Language Models Only, 2021, <https://arxiv.org/abs/2110.05448>.
- [12] Y. Li et al., Competition-Level Code Generation with AlphaCode, 2022, <https://arxiv.org/abs/2203.07814>.
- [13] R. Nakano et al., WebGPT: Browser-assisted question-answering with human feedback, 2021, <https://arxiv.org/abs/2112.09332>.
- [14] A. Lewkowycz et al., Solving Quantitative Reasoning Problems with Language Models, 2022, <https://arxiv.org/abs/2206.14858>.

### profile

前原 義明 (まえはら よしあき)

審査第二部 一般機械 (制動・機械要素) 審査官

平成22年4月 入庁 (特許審査第二部 自動制御)

平成25年4月 審査官昇任 (審査第二部 自動制御)

平成27年10月 総務部企画調査課

平成28年10月 審査第二部 運輸

平成30年1月 文部科学省 宇宙関係在外研究員 (ドイツ航空宇宙センター)

平成31年1月 審査第二部 自動制御 (流体制御)

平成31年4月 審査第二部 審査調査室

令和元年10月 審査第二部 生活機器

令和2年7月 一般財団法人 日本特許情報機構 知財AI研究センター

令和4年7月 現職