



ChatGPT 入門

かわさきしんじ , Deep Insider 編集部 [著]

01.ChatGPT とは何か そのできること／できないこと

02.ChatGPT や InstructGPT はなぜユーザーの意図に沿った返答を生成できるのか？

03.OpenAI Cookbook で学ぶ ChatGPT プロンプトの基礎の基礎

04. 新しい Bing に組み込まれた ChatGPT よりも強力な言語モデルに触ってみよう

05. 思考の連鎖（Chain of Thought）で ChatGPT からよりよい応答を引き出そう

06.ChatGPT の API を使ってみよう：コンソールで対話するコードとは？

ChatGPT とは何か そのできること／できないこと

OpenAI がリリースした人との対話をターゲットとする大規模な言語モデル「ChatGPT」。その概要について見てみましょう。

かわさきしんじ, Deep Insider 編集部 (2022 年 12 月 23 日)

ChatGPT とは

ChatGPT は 2022 年 11 月末に OpenAI がリリースした対話に特化した言語モデル（と、このモデルを使ってユーザーが対話するための Web サービス）です。

The screenshot shows the ChatGPT web interface. On the left, a sidebar menu includes 'New Chat' (highlighted), 'ChatGPT Operation Exp' (with a dropdown arrow), '殺人毒薬作り禁止', '2022年のサッカーワールドカップ優', and 'Pythonでパスからディレクトリ名とフ'. Below these are 'Dark Mode', 'OpenAI Discord', 'Updates & FAQ', and 'Log out'. The main area shows a conversation history with the user asking 'ChatGPTの動作原理を教えてください' and the AI responding about its training process, text data used, architecture, conversational abilities, and its role as a GPT model. A 'Regenerate response' button is at the bottom right of the message input field. A footer note at the bottom states 'ChatGPT Dec 15 Version. Free Research Preview. Our goal is to make AI systems more natural and safe to interact with. Your feedback will help us improve.'

ChatGPT

流れのように自然な会話ができたり、ときには素っ頓狂（すっとんきょう）な答えが返ってきたりするのが楽しくて、このサービスを使っている方もたくさんいるでしょう。本稿では、その特徴や実際に使った例などを見ながら、ChatGPT がどんなものかを概観することにします。

特徴

ChatGPT はテキスト生成用に訓練された GPT-3.5 と呼ばれる系列の言語モデルを対話に適したモデルへとファインチューンしたものです。このときには、RLHF (Reinforcement Learning with Human Feedback)。人間のフィードバックを用いた強化学習) と呼ばれる手法が使われています（この手法の概要については次回以降に紹介する予定です）。

ChatGPT を使う上で気になるのは使用料がかかるかですが、現在は初期の Research Preview であり、無料で使用できます（ということは、将来的には有料になるかもしれませんね）。

なぜこうも ChatGPT との対話が自然なのかといえば、その学習ではインターネット上のテキストを用いており、その中には対話テキストも含まれているからだそうです。

会話が自然なことにはもう一つ理由があります。それはユーザーが入力した内容をモデルが覚えていたり (ChatGPT ではユーザーが話した内容を 3000 語あるいは 4000 トークンまでさかのぼって参照可能)、モデルから出力された内容に間違いがあったときに、ユーザーがそれを訂正したりできることです。人間同士の会話でも「アレってソレだっけ?」「いやコレだよ」みたいなことはよくあります。それと似たことを実現しているので、会話が自然に感じられるかもしれません。同時に、これはモデルと会話をしながら、特定の話題に関する知識を積み上げていき、何らかの結論に達することもできるということです。

ChatGPT は非常に多くのテキストを用いて学習されているので、その知識量は膨大な物ですが、常にユーザーの入力に対して間違った返答を返したり、他者を害するような内容などを返答したりすることには注意が必要です。また、学習した内容に含まれている事象は 2021 年末までのものであることにも注意しましょう。例えば 2022 年のサッカーワールドカップでどこが優勝したかなんて情報はこのモデルは知りません。



トンチンカンな返答が楽しいというのもあるかもしれませんね（かわさき）。



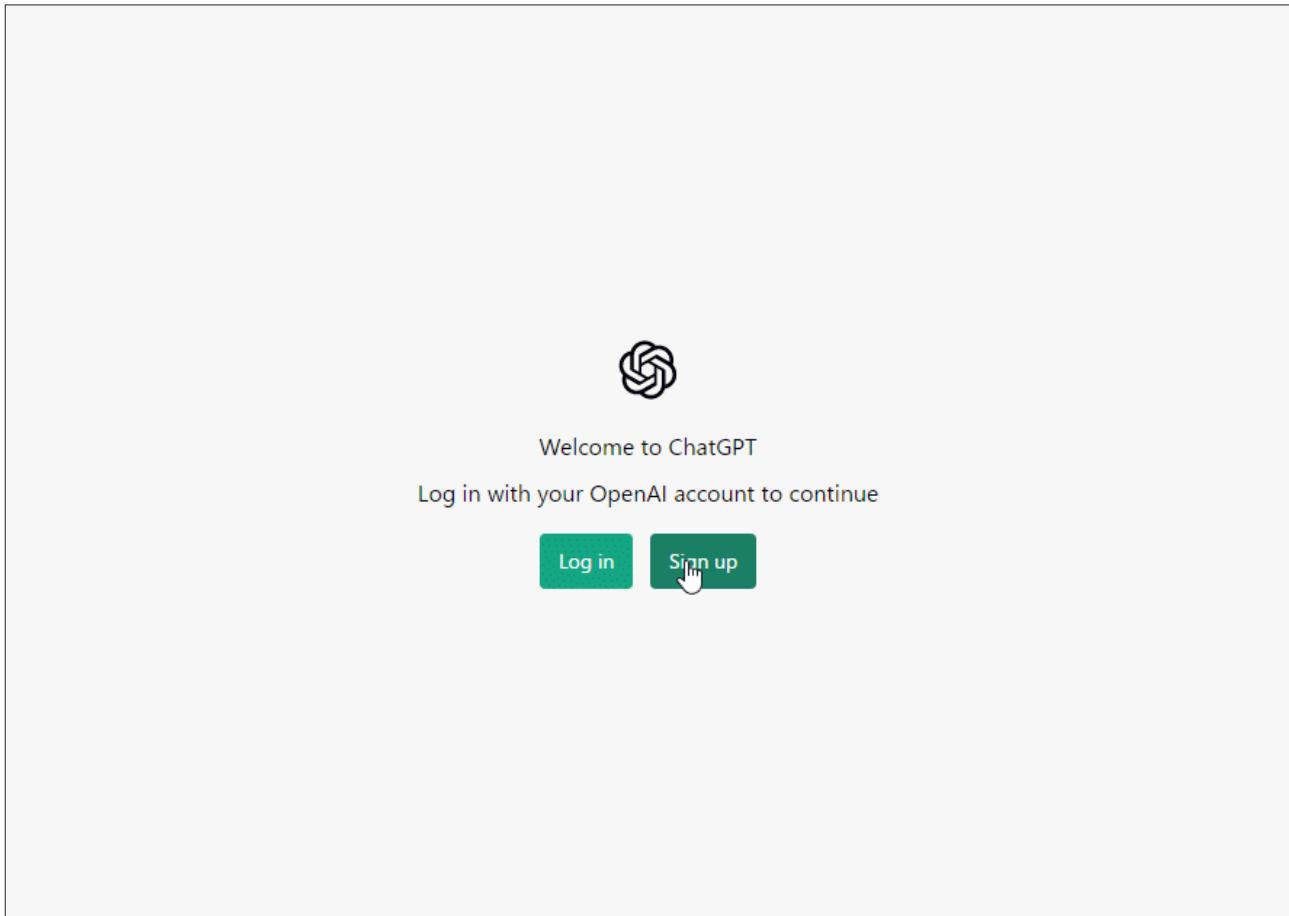
「機械学習とは何ですか？」と質問すると非常に良質で短く分かりやすい回答が得られたり、「Python でブロック崩しのゲームを作つて。」と質問すると本当に動作するコードと作り方が回答されたりします。これで「すごい」となるのですけど、「とんこつラーメンの作り方を教えて。」と質問すると、自信満々に理路整然と「おしるこを入れ」たり「スキマ棒で泡を取り除い」たりとんでもないウソを教えてくれます。というかスキマ棒って何……。すごく正確と思えるときと自信満々にウソのときで回答の仕方が同じなので、全てを素直に信じられないのが残念なところですね（一色）。

ChatGPT を試してみよう

以下では、ChatGPT にサインアップし、幾つかの質問をして、これがどんなものかを簡単に見てみましょう。

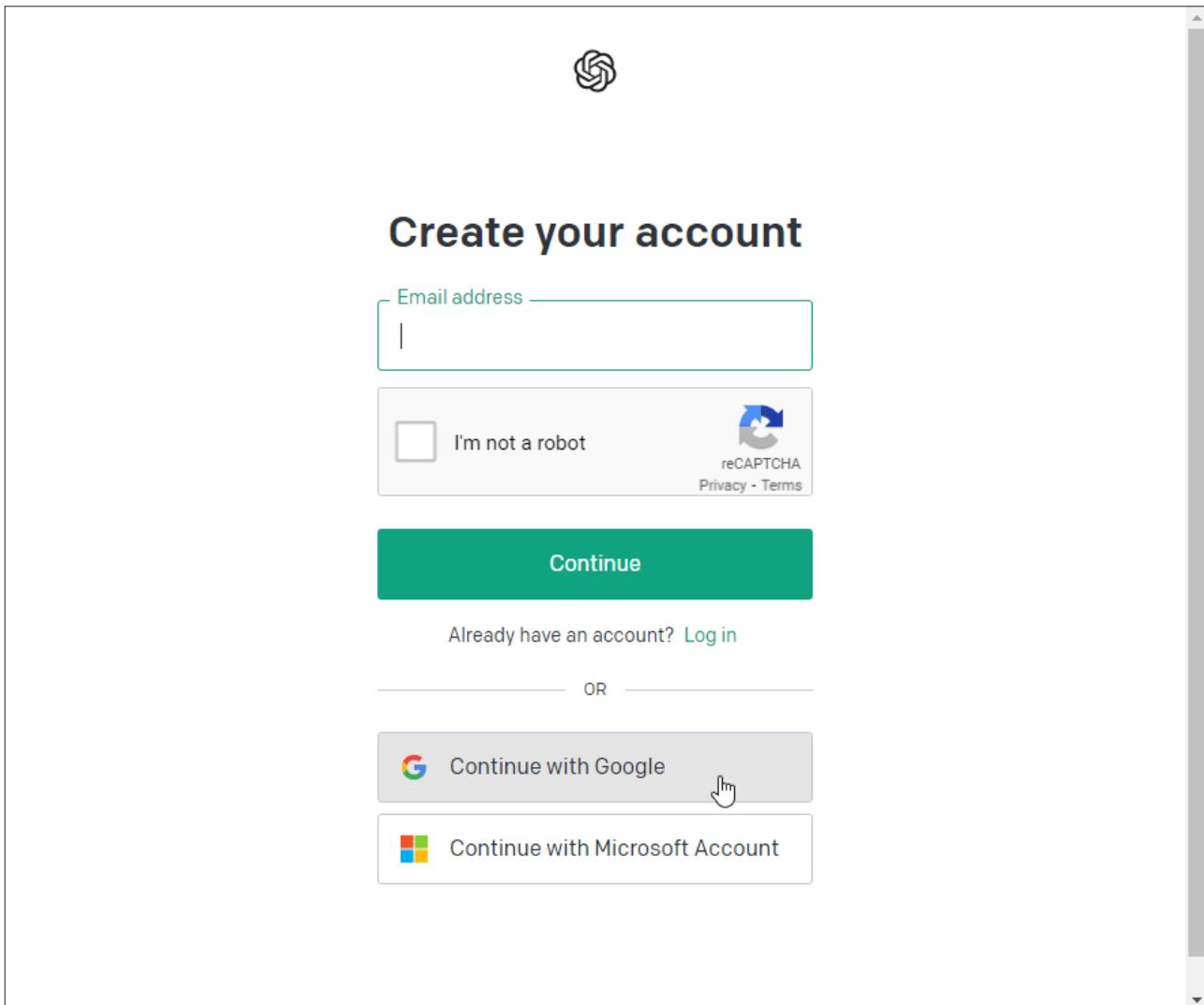
サインアップ

ChatGPT を試してみるには、サインアップが必須です。ログインしていない状態で [ChatGPT のページ](#)にアクセスすると以下の画面が表示されます。



ChatGPT のログイン／サインアップページ

ここで [Sign up] ボタンをクリックすると以下のような画面になります。



サインアップに使うアカウントを指定（ここでは Google アカウントを使用）

ここでは [Continue with Google] ボタンをクリックして、Google アカウントでのサインアップをしました。これにより使用する Google アカウントの選択画面になるので、使用するアカウントを選択し、Google アカウントのパスワードを入力します（2段階認証を利用している場合は、何らかの手段で 2段階認証が行われます）。

認証が通ると、次のような画面が表示され、名前と電話番号を入力する画面が順次表示されるので情報を入力しましょう。



Tell us about you

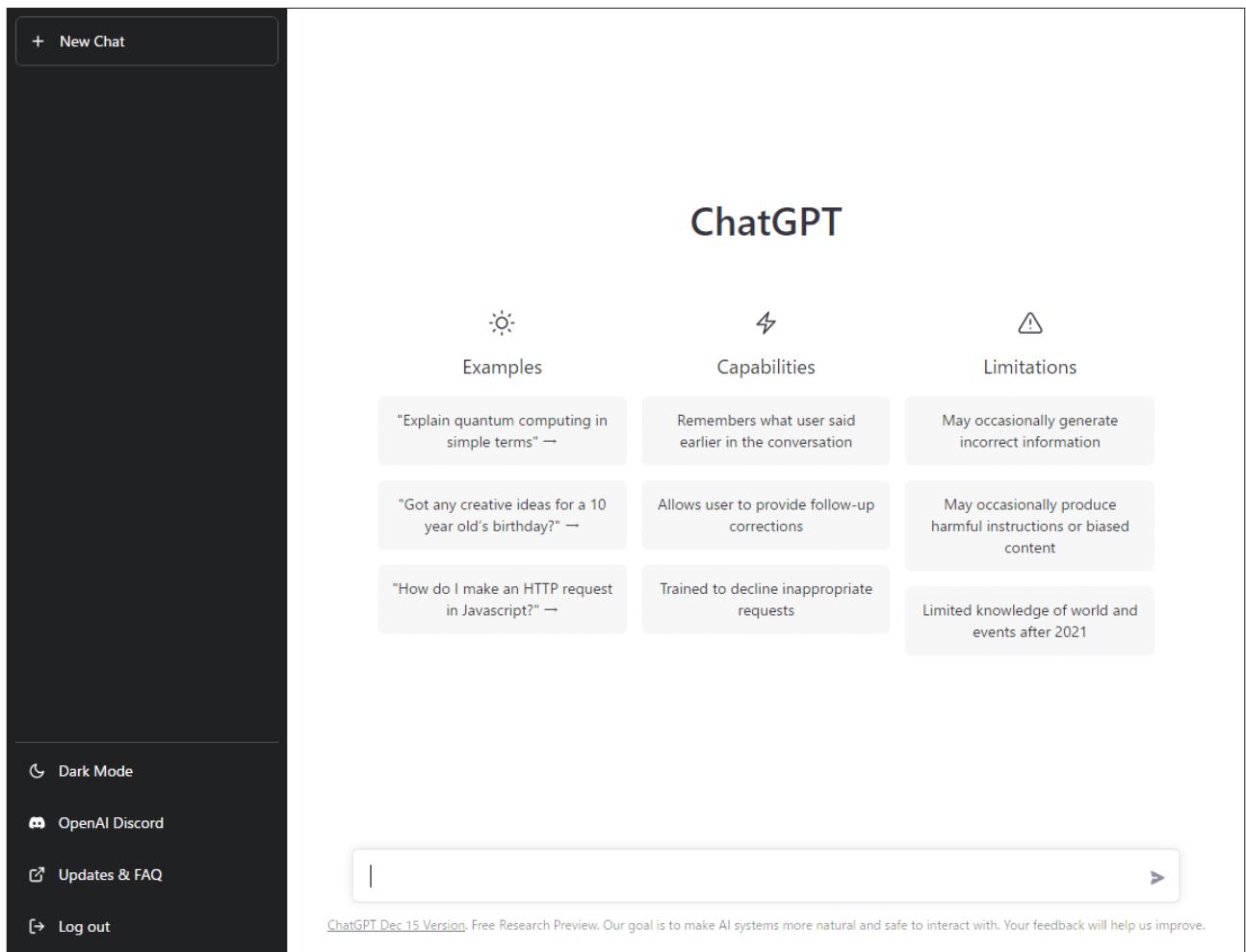
Shinji Kawasaki

Continue

By clicking "Continue", you agree to our [Terms](#) and confirm you're 18 years or older.

名前の入力ページ

その後、ChatGPT の特徴を説明したダイアログが表示されます。ここで [Next] ボタンや [Done] ボタンを押していくけば、次のように ChatGPT にプロンプトを入力可能な画面になります。



ChatGPTとの対話を行う画面

ウィンドウの右側には入力例／ChatGPT の機能／制限が表示され、その下にプロンプトの入力ボックスがあります。一方、左側の一番上には [New Chat] ボタンがあります。これを使うと、ChatGPT との対話を行うセッションを新たに作成できます。セッションはこのボタンの下に並べられます。

また、その下には4つのボタンがあります。[Dark Mode] ボタンは対話部分の背景を黒に変更します。[OpenAI Discord] ボタンは Discord の OpenAI チャネルを開くためのものです。[Updates & FAQ] は ChatGPT に関するアップデート情報や FAQ を一覧するページを開きます。[Log out] は ChatGPT からログアウトするボタンです。



日本語の入力も受け付けて、日本語で返答してくれるのはうれしいですね。

Pythonについて質問してみる

ChatGPTはプログラミングの分野の知識も豊富です。そこでまずは「Pythonでパスからディレクトリ名とファイル名を取得するには?」と入力してみましょう。以下はその結果です（結果は試すごとに内容が少し変化するかもしれません）。

The screenshot shows the ChatGPT interface. On the left, there's a sidebar with options like 'New Chat', 'Dark Mode', 'OpenAI Discord', 'Updates & FAQ', and 'Log out'. The main area has a search bar with the query 'Pythonでパスからディレクトリ名とファイル名を取得するには?'.

Response:

Pythonには、「os.path」モジュールがあります。このモジュールには、パスから情報を取得するためのいくつかの関数があります。

パスからディレクトリ名を取得するには、「os.path.dirname」関数を使用します。この関数は、指定したパスのディレクトリ部分を返します。たとえば、次のように使用できます。

```
import os

path = '/home/user/documents/files/example.txt'
dirname = os.path.dirname(path)
print(dirname) # '/home/user/documents/files'
```

パスからファイル名を取得するには、「os.path.basename」関数を使用します。この関数は、指定したパスのファイル名部分を返します。たとえば、次のように使用できます。

```
import os

path = '/home/user/documents/files/example.txt'
filename = os.path.basename(path)
print(filename) # 'example.txt'
```

At the bottom, there's a note: 「注意: 「OpenAI AI モデル」は、Docker 容器内の Python 基本ライブラリが「py�t」フォルダにトマト」 and a 'Regenerate response' button.

os.pathモジュールを使った解説

うん。合っていますね。[Copy code]ボタンがあるからコードを実際に試してみるのも簡単です（実際に実行してみました）。

実は、これと同じ内容は、本フォーラムの解決！Pythonという連載の「パスからディレクトリ名とファイル名を取得するには？」でも紹介しています。



プロンプトはあえて、解決！Pythonの記事と同じようにしています。

ただし、記事の方ではpathlibモジュールを使う例も紹介しているので、記事の方がエラそうな気はします。

そこで「pathlib モジュールを使うとどうなりますか？」と入力してみます。

The screenshot shows a ChatGPT interface. The user has asked, "pathlibモジュールを使うとどうなりますか？". The AI response explains that the `pathlib` module provides a more direct and object-oriented way to handle file paths compared to the `os.path` module. It includes a code example:

```
from pathlib import Path

path = '/home/user/documents/files/example.txt'
dirname = Path(path).parent
print(dirname) # PosixPath('/home/user/documents/files')
```

Below the code, the AI notes that using the `name` attribute of a `Path` object will return the file name. A second code block is shown, which is identical to the first but lacks the explanatory text.

At the bottom of the interface, there are links for "Dark Mode", "OpenAI Discord", "Updates & FAQ", and "Log out". A footer bar at the bottom of the window displays the text "ChatGPT Dec 15 Version. Free Research Preview. Our goal is to make AI systems more natural and safe to interact with. Your feedback will help us improve."

pathlib モジュールを使った場合の返答

少々長い説明の後、正しい使い方が紹介されました（ただし、上の画像では、ディレクトリ名を `print` 関数で表示した結果をコメントとして記述しているところは間違えています）。

これを見ると、筆者は少し怖くなってしまいます。解決！Pythonは主にGoogleなどの検索サイトで「Pythonパスディレクトリ名」などとして検索して、記事を読みにきてもらうことを念頭に置いた連載です。ところが、ChatGPTとお話をすれば、言語モデルがその使い方をやさしく教えてくれるのです。「あれ？ 記事を書く必要がなくなっちゃう？」となりますよね。

Google Chrome 用の ChatGPT 拡張機能

世の中には目の付け所がよい人がいて、Google で検索した語を ChatGPT に渡して、その結果を検索結果ページに表示してくれる [Chrome / Firefox 用の拡張機能](#)も既に存在しています（ただ、この拡張機能を使うと、自分が OpenAI にログインしているアカウントで ChatGPT のセッションが新規に作成され、その結果、ページ左側に各セッションを表す [New Chat] などのタイトルが付いたタブが多数表示されてしまうかもしれませんのでご注意ください）。

The screenshot shows a Google search results page for the query "Python パスディレクトリ名". The results list several articles and blog posts from various sources. On the right side of the page, there is a vertical sidebar for "ChatGPT". The sidebar has a title "ChatGPT" and contains text explaining how to use the `os.path` module to get the directory name from a path string. It includes examples of Python code:

```
import os
path = '/path/to/dir/file.txt'
dirname = os.path.dirname(path)
print(dirname) # '/path/to/dir'
```

Below this, it says "この例では、path 変数に格納されているパスのディレクトリ名を取得しています。もし、ディレクトリ名だけを取得したい場合は、os.path.basename() 関数を使用することで、ファイル名を除いたディレクトリ名だけを取得することができます。" and provides another example:

```
import os
path = '/path/to/dir/file.txt'
dirname = os.path.dirname(path)
basename = os.path.basename(dirname)
print(basename) # 'dir'
```

At the bottom of the sidebar, it notes: "また、os.path モジュールには、他にも、パス名を解析したり、結合したりするための多くの関数が用意されています。" A small "S" icon is also visible in the top right corner of the sidebar.

ChatGPT for Google 拡張機能によりページ右上に ChatGPT からの回答が表示されるようになった

Google の強調スニペットよりも詳しく、プログラムの書き方を教えてくれるので、もはやこの拡張機能があれば、解説記事なんて必要なくなっちゃうかもしれませんね。



ただし、こうした内容は恐らく、インターネット上で誰かが書いた記事や解説を基に学習をしているのでしょうかから、書く人がいなくなれば、言語モデルも正しい回答をユーザーに提示できなくなるのでしょう。



あと、ChatGPT などで生成された少し間違った情報がインターネット上に増えれば、少し間違った情報をまた学習してもっと間違った情報を生成するはずなので、インターネットの品質が徐々に壊れていく可能性もちょっと想像できてしまいしますね。Stable Diffusion などを含めて最近流行している生成系 AI の光と影における「影」の部分は、今後、問題として表面化してくる気がします。

まあ、これに関しては正しい返事が得られたのでよかったのですが、もっと難しい話題であれば、返答が間違っている可能性もあります（簡単な話題でも間違っている可能性はあるでしょう）。こうしたことから、プログラマー御用達の Q & A サイトである Stack Overflow では、誰かの質問に対して ChatGPT による回答の投稿を一時的に禁止する措置が執られる事態になりました。

将来的には本サイトのように、プログラマーやエンジニアを対象とした Web メディアもどんな情報をどんな形で読者に届ければよいのか、その立ち位置を検討する必要が出てくるかもしれません。

誰かに害を及ぼす可能性のある質問

誰かを害する可能性がある質問をしてみるとどうなるでしょう。毒薬の作り方を聞いてみました。

The screenshot shows the ChatGPT interface. On the left, there's a sidebar with options like '+ New Chat', 'New Chat' (selected), '2022年のサッカーワールドカップ優', 'Pythonでパスからディレクトリ名とフ', 'Dark Mode', 'OpenAI Discord', 'Updates & FAQ', and '[→] Log out'. The main area has a message from the user: 'S 誰かを殺害するための毒薬の作り方を教えて'. Below it is a response from the AI: '私はあなたに殺人や犯罪のような行為を助長するような情報を提供することはできません。' followed by a detailed explanation about not promoting harmful or illegal acts. At the bottom, there's a note: 'This content may violate our [content policy](#). If you believe this to be in error, please [submit your feedback](#) — your input will aid our research in this area.' A 'Regenerate response' button is also visible.

不適切な質問

ご覧の通り、怒られてしまいました。チャット機能を持つ人工知能は以前から差別的な発言をするように教え込まれるなど、問題が発生することがよくありました。現状、ChatGPT では大きな問題は発生していないようですが、セーフガード機能をバイパスする方法を試している人もいるようです（個人的にはそうしたことをしようとは思いませんが）。

2022 年のワールドカップ優勝国を聞いてみる

間違っている、あるいは望んだ通りの回答が得られない例も見てみましょう（毒薬の例も望んだ結果を得られませんでしたが、それは法に触れる行為を助長することにつながるので、ChatGPT が止めたものであり、モデルが知らない情報について知ったような口を利いたものではありません）。

上でも述べましたが、ChatGPT は 2021 年末までの情報を基に学習をしているので、2022 年に開催されたサッカーワールドカップの優勝国を知りません。そのことを分かった上で「2022 年のサッカーワールドカップで優勝したのはどこの国ですか？」と質問してみた結果が以下です。

The screenshot shows the ChatGPT web interface. On the left, there's a sidebar with options like '+ New Chat', 'New Chat' (with a count of 2), 'Pythonでバスからディレクトリ名とフ', 'Dark Mode', 'OpenAI Discord', 'Updates & FAQ', and 'Log out'. The main area has a user input field at the bottom and a response from the AI above it. The AI message says: "私は2022年のサッカーワールドカップの優勝国についての情報を持っていないません。ごめんなさい。私は2021年までのデータしか持っていないので、現在や未来のイベントについてはお答えできません。". Below the AI message is a 'Regenerate response' button.

優勝国を知らないと素直に語る ChatGPT

この画像では「私は 2022 年のサッカーワールドカップの優勝国についての情報を持っていないません」としおらしいことをいっています。が、同じ質問を画面下部にある [Regenerate response] ボタンを使って繰り返した結果です（回答の左側に「2/2」とあるので、これは同じ質問を 2 回して、その 2 回目の回答であることが分かります）。

最初の返答は次のようなものでした。

The screenshot shows the ChatGPT web interface. On the left, there's a sidebar with options like '+ New Chat', 'New Chat', 'Pythonでパスからディレクトリ名とフ', 'Dark Mode', 'OpenAI Discord', 'Updates & FAQ', and '[→ Log out]'. The main area has a dark header bar with a 'S' icon and the question '2022年のサッカーワールドカップで優勝したのはどの国ですか？'. Below the question, it says '現在、2022年のサッカーワールドカップは開催されておりません。サッカーワールドカップは4年に1度、FIFA（国際サッカー連盟）が主催する大会です。次のサッカーワールドカップは2026年にアメリカ、カナダ、メキシコで開催されます。そのため、2022年のサッカーワールドカップの優勝国はまだ決まっていません。' At the bottom, there's a 'Regenerate response' button and a note about the ChatGPT Dec 15 Version.

最初の回答

「現在、2022 年のサッカーワールドカップは開催されておりません」とあるので、この返答は明らかに間違っています。ChatGPT はあくまでも学習した内容を基にそれらしいテキストを生成するものだという認識を常に持ち、その回答を鵜呑みにしないで、正しいかどうかを確認する必要があります。

ChatGPT は現在のところ、正確な情報を教えてくれることもあれば、不正確な情報を返すこともあります。このことを頭に留め置いて、その可能性を模索するというのがよさそうです。次回以降ではさまざまな情報を基に、これがどのようにして動作しているかを考えてみる予定です。

ChatGPT や InstructGPT はなぜユーザーの意図に沿った返答を生成できるのか？

ChatGPT やその前身ともいえる InstructGPT は、GPT とは異なる目的を持ったモデルです。それ故にこれまでとは異なり、ユーザーの意図に沿ったテキストを生成できます。その違いを見てみましょう。

かわさきしんじ, Deep Insider 編集部 (2023 年 01 月 13 日)

GPT、InstructGPT、そして ChatGPT

ChatGPT はその名の通り、対話に特化した言語モデルです。GPT 3 (GPT 3.5) をベースとしていますが、GPT 3 から ChatGPT が生まれるまでの間にはもう一つ重要な言語モデルがあります。それが InstructGPT です (InstructGPT 自体は GPT 3 をベースとしているようです)。

では、GPT → InstructGPT → ChatGPT という進化がなぜ起きたのでしょうか。InstructGPT についての論文ではその概要でおおよそ次のようなことが述べられています。つまり、「大規模な言語モデルは嘘、有害な出力を生成したり、単にユーザーの役には立たない出力を生成したりする。言い換えれば、これらのモデルはユーザーに合ったものになっていない (not aligned with their users)」ということです。

これは GPT のような大規模言語モデルが目的としているのは「一連のトークン（単語）が入力されたときに、次のトークンは何かを予測する」ことであり、「ユーザーの指示に従って有用で無害な出力を行う」ことではないからです。InstructGPT はまさにユーザーの指示 (instruction) に従った出力が行えるように GPT をチューンしたもので、ChatGPT は InstructGPT をベースに対話を行えるようにチューンしたものだと考えることができるでしょう。

モデル	目的
GPT	入力されたトークンを基に次に出現するトークンを予測する
InstructGPT	ユーザーの指示に従って有用で無害な出力を行う
ChatGPT	InstructGPTと同様な学習方法を用いて対話に特化した出力を行う

GPT / InstructGPT / ChatGPT の違い

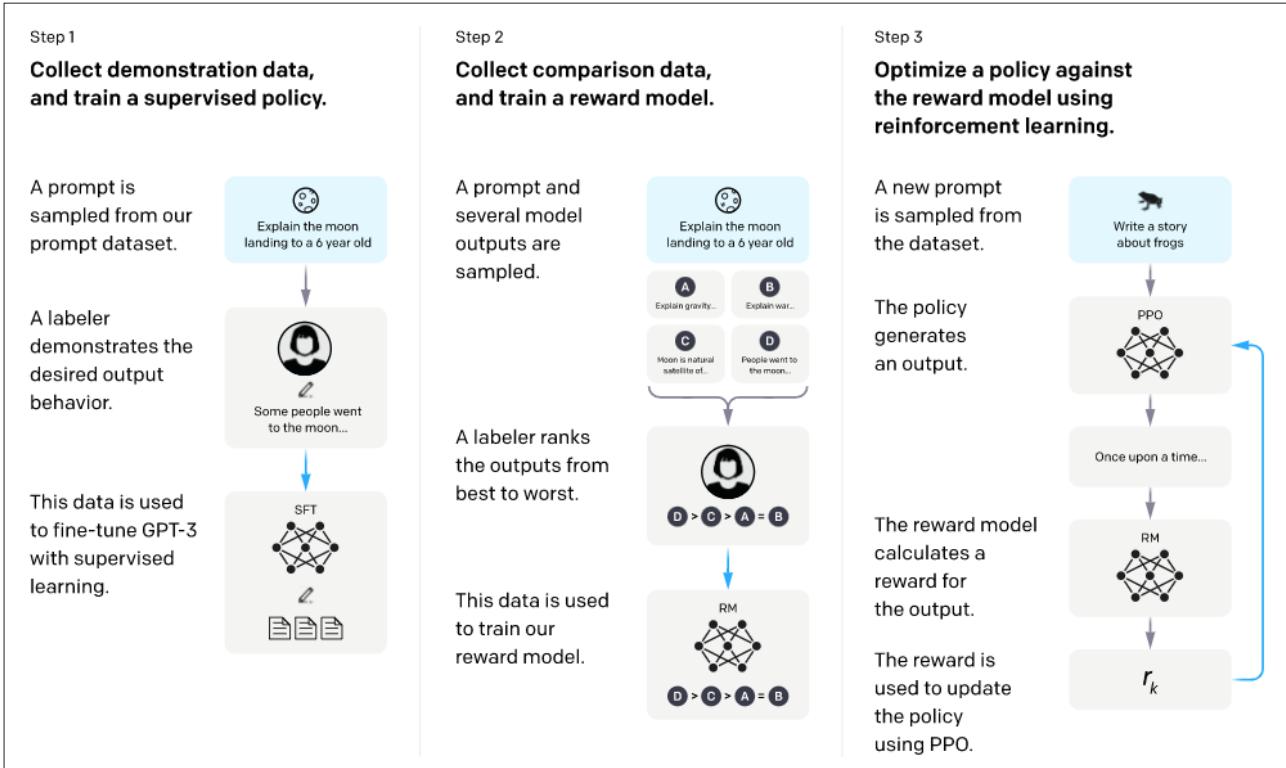
有用で無害な出力を得られるようにするために、ChatGPT と InstructGPT ではそれらの訓練過程で RLHF (Reinforcement Learning from Human Feedback、人間のフィードバックを基にした強化学習) と呼ばれる手法が採用されているのが重要です（ただし、ChatGPT と InstructGPT とではデータ収集の方法に違いがあり、これが ChatGPT を対話に特化したものとしていると思われます）。



やばい。コードを少しは出そうと思っていたんですが、この回、コードが出てくるのかなあ（かわさき）。

RLHF (Reinforcement Learning from Human Feedback)

では、この RLHF とはどんなものでしょうか。以下に OpenAI のブログ記事からその手法を説明する図を引用します。



InstructGPT における RLHF

『Aligning Language Models to Follow Instructions』より引用。

図の内容を文章でまとめると次のような感じになるでしょうか。

- ・ステップ 1：あるプロンプトに対する望ましいモデルからの出力を人間が用意して、GPT モデルを教師あり学習でファインチューンする
- ・ステップ 2：何らかのプロンプトに対するステップ 1 のモデルの出力を幾つかサンプリングして、出力にランク付けをし、そのデータを使って報酬モデルの学習を行う
- ・ステップ 3：報酬モデルを使って GPT モデルの強化学習を行う

ステップ 1 では、以前のバージョンの InstructGPT に入力されたプロンプトをデータセットとして、その一部を取り出し、それらのプロンプトが入力されたときにモデルがどう振る舞えばよいか（どんな出力を行えばよいか）、その出力として望ましいものを人間が用意してやります。そして、「プロンプトと望ましい出力」の組み合わせを使って教師あり学習を行って、GPT モデルをファインチューンします (Supervised Fine-Tune、SFT)。

ステップ 2 では、このモデルに対して何らかのプロンプトを入力し、そこから幾つかの出力を得た上で、どの出力が望ましいか／望ましくないか、人間がランク付けをします。そして、そのデータを使い報酬モデルの学習を行います。

ステップ 3 では、GPT モデルを強化学習します。このときにはステップ 2 で作成した報酬モデルが使われます。

このうち、ステップ 2 とステップ 3 を繰り返すことで、プロンプトに対する出力にランク付けを行ったデータが新しく得られ、その結果、報酬モデルが更新され、強化学習もさらに進むといった具合に学習が行われます。



このように学習の過程に人間を組み込んだものを「ヒューマン・イン・ザ・ループ」と呼ぶことがありますね。

このようにして作成されたのが InstructGPT です。ChatGPT は対話に特化したモデルであることから、学習に使われるデータの収集方法が少し異なっています。

「ChatGPT: Optimizing Language Models for Dialogue」によれば、「ユーザーと AI との対話を、人間（AI トレーナー）が両方の立場でデモしたデータ」がステップ 1 のデータとして使われます。これらが InstructGPT の学習で使われたデータセットと統合され、対話の形式に変換されたものが GPT モデルのファインチューン（SFT）に使われます。さらにステップ 2 では AI トレーナーとチャットボットとの対話を基にデータ収集（とランク付け）を行います。これらが InstructGPT と ChatGPT のデータ収集方法の違いです。

InstructGPT にしても ChatGPT にしても、重要なのは報酬モデルの学習に使われるデータのランク付けでしょう。モデルからの出力の品質に応じてランク付けを行うことで、人にとって有用で害のない出力を行えば、より多くの報酬をもらえるようにモデルが学習することでモデルが生成するテキストがユーザーの意図や指示に沿ったものになるのです。

さらにいえば、InstructGPT ではユーザーからの入力（プロンプト）に対してモデルがどういう出力を返せばよいのかを学習させることで単に次に出力されるトークン（単語）を推測するというモデルではなく、ユーザーの指示に合った出力を推測するようなモデルになっています。そして、ChatGPT では対話形式のデータセットを使うことで、それを対話に特化したものにできているということです。これが ChatGPT（や InstructGPT）からの出力が人間にとて好ましい出力となっている大きな理由なのでしょう。



これまで強化学習は自動的なゲーム操作や自動走行などに使われていてあまり目立っていないかったので、こんなふうに使われて面白いですね（一色）。

さまざまなモデル

OpenAI のドキュメント「[Model index for researchers](#)」には GPT や InstructGPT に関するモデルについての説明があります（残念ながら ChatGPT についてはまだ記載がないようです）。

これによれば InstructGPT に関するモデルには以下のようなものがあります。

モデル	説明
code-davinci-002	コード生成に適したモデル。text-davinci-002のベース
text-davinci-002	InstructGPTモデル。code-davinci-002がベース
text-davinci-003	text-davinci-002を改善したモデル

InstructGPT に関するモデル

他のモデルについては上記のリンク先を参照してください。

Web で InstructGPT を試すには OpenAI が用意している [Playground ページ](#) が使えます（OpenAI へのサイドアップ／ログインが必要です）。

The screenshot shows the OpenAI Playground interface. On the left, there's a sidebar titled "Get started" with instructions about using the API and sharing requests. The main area is titled "Playground" and contains a text input field with the placeholder "晴れた日曜日の午後には何をすればいいかな？". To the right of the input field are several configuration sliders and dropdowns: Mode (set to text-davinci-003), Temperature (0.7), Maximum length (256), Stop sequences (empty), Top P (1), Frequency penalty (0), Presence penalty (0), Best of (1), Inject start text (checked), and Inject restart text (unchecked). At the bottom, there are "Submit" and "Try it now" buttons, and a status bar showing "223". A small pop-up window at the bottom says "Looking for ChatGPT? Try it now".

Playground ページ

上の画像を見れば分かる通り、中央の大きな区画に InstructGPT への入力とそれに対する出力が表示されます（薄緑の背景色のテキストが InstructGPT からの出力です）。また、右上の [Model] 欄には「text-davinci-003」と表示されているので、InstructGPT の改善版モデルがここでは使われていることが分かります。



中央の大きな区画の下部には「Looking for ChatGPT?」であることから、ここで使われているモデルが ChatGPT ではないということも想像できますね。

コードからモデルを使うには

では、コードから簡単に InstructGPT モデルを使ってみましょう。OpenAI は InstructGPT の API を公開しているのでこれを呼び出すだけのホントに簡単なコードです（ノートブックは[こちら](#)）。

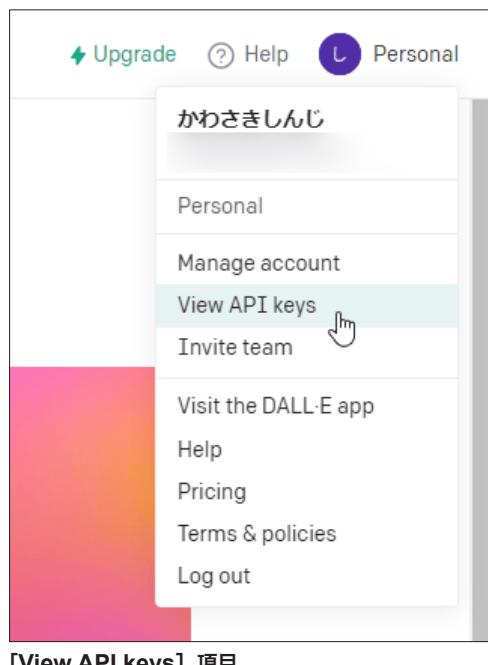


取りあえずコードを出してみることにしました（笑）。

そのために必要な手順は以下です。

1. OpenAI にサインアップ／ログインして、API キーを取得する
2. PyPI から [openai モジュール](#)をインストールする
3. openai.Completion.create クラスメソッドを呼び出す

API キーは OpenAI にサインアップ／ログインした後に、右上のアカウントアイコンをクリックすると表示されるメニューから [View API keys] を選択します。



[View API keys] 項目

すると、以下のようなページが表示されるので、[Create new secret key] ボタンをクリックしてください。

API keys

Your secret API keys are listed below. Please note that we do not display your secret API keys again after you generate them.

Do not share your API key with others, or expose it in the browser or other client-side code. In order to protect the security of your account, OpenAI may also automatically rotate any API key that we've found has leaked publicly.

SECRET KEY	CREATED	LAST USED
sk-...uSdu	2023年1月10日	Never

[+ Create new secret key](#)

Default organization

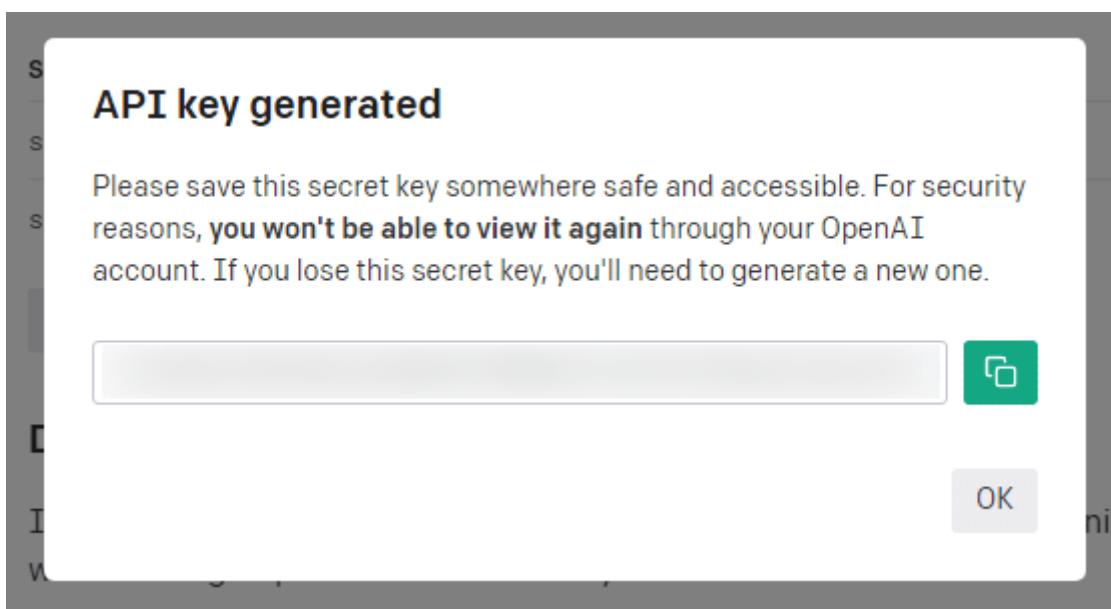
If you belong to multiple organizations, this setting controls which organization is used by default when making requests with the API keys above.

Personal

Note: You can also specify which organization to use for each API request. See [Authentication](#) to learn more.

[Create new secret key] ボタンをクリック

これで API キーが作成され、次のようなダイアログにキーが表示されます。このキーは一度しか表示されないので、忘れずにコピーしておくようにしましょう。



生成された API キー

そしてシェルやコマンドプロンプト、あるいは Visual Studio Code などの開発環境のシェルやコマンドプロンプトから「`pip install openai`」コマンドを実行するか、Jupyter ノートブックのセルで「`!pip install openai`」を実行します。

これで準備は完了です。後は `openai` モジュールをインポートし、その `api_key` 属性に、先ほど生成された API キーを代入し、最後に `openai.Completion.create` クラスメソッドを呼び出すだけです。以下に例を示します。

```
KEY = '取得したAPIキーをここに記述'

import openai

openai.api_key = KEY

response = openai.Completion.create(
    model='text-davinci-003', # InstructGPT
    prompt='晴れた日曜日の午後には何をすればいいかな?',
    temperature=0.7,
    max_tokens=256,
    top_p=1,
    frequency_penalty=0,
    presence_penalty=0
)

print(response['choices'][0]['text'])
```

InstructGPT の API を呼び出すコード

InstructGPT の API を呼び出すので、`model` 引数には '`text-davinci-003`' を指定します。`prompt` には先ほど Web で試したのと同じ '晴れた日曜日の午後には何をすればいいかな?' を指定しました。その他の引数については「[Create completion](#)」を参照してください。

実行した結果を以下に示します。

実行結果

中身のことはよく分からなくても、APIを呼び出せばそれなりの答えが返ってくるというのは楽でいいですね(笑)。



これは簡単ですね。Python などでソフトウェアのプログラムを書ける人は多いと思うので、多くの人が試すことで思いもしない面白い活用方法やソリューションのアイデアもたくさん出てきそうだと思います。

というわけで、今回は InstructGPT と ChatGPT という 2 つのモデルがどんなふうに作られているかを見てきました。次回は何をするか未定ですが、何か面白いことをできたらと思っています。

OpenAI Cookbook で学ぶ ChatGPT プロンプトの基礎の基礎

OpenAI が提供している Cookbook では大規模言語モデルからの出力をどうすればよいものにできるか、そのノウハウが紹介されています。その基本部分を見てみましょう。

かわさきしんじ, Deep Insider 編集部 (2023 年 02 月 03 日)



筆者がネタに苦しんでいるうちに、マイクロソフトが Azure に ChatGPT を含む OpenAI のサービスを採用すると発表したり、ChatGPT Professional のウェイトリストが用意されたりと、世の情勢は活発にうごいていますね（かわさき）。

OpenAI Cookbook

OpenAI は「[OpenAI Cookbook](#)」と呼ばれるリポジトリを GitHub で公開しています。これは、OpenAI が提供する API を使って何らかのタスクを行うためのサンプルコードやガイドを示したものです。今回はそのうちの GPT 3 に関する内容を幾つか紹介しましょう。ただし、OpenAI Cookbook で紹介されているノウハウは ChatGPT に特化して書かれているわけではないことには注意してください。

The screenshot shows the GitHub repository for the OpenAI Cookbook. The README.md file contains introductory text about the cookbook, instructions for running examples, and information that most code examples are written in Python. Below the README, there's a section titled "Guides & examples" with a list of various topics and their sub-examples, all of which are in Python. To the right of the repository view, there's a sidebar showing a "Languages" chart indicating that 100.0% of the code is in Python.

README.md

OpenAI Cookbook

The OpenAI Cookbook shares example code for accomplishing common tasks with the OpenAI API.

To run these examples, you'll need an OpenAI account and associated API key ([create a free account](#)).

Most code examples are written in Python, though the concepts can be applied in any language.

Guides & examples

- API usage
 - [How to handle rate limits](#)
 - [Example parallel processing script that avoids hitting rate limits](#)
 - [How to count tokens with tiktoken](#)
 - [How to stream completions](#)
 - GPT-3
 - [Guide: How to work with large language models](#)
 - [Guide: Techniques to improve reliability](#)
 - [How to use a multi-step prompt to write unit tests](#)
 - [Text writing examples](#)
 - [Text explanation examples](#)
 - [Text editing examples](#)
 - [Code writing examples](#)
 - [Code explanation examples](#)
 - [Code editing examples](#)
 - Embeddings
 - [Text comparison examples](#)
 - [How to get embeddings](#)
 - [Question answering using embeddings](#)

Languages

Python 100.0%

OpenAI Cookbook

この Cookbook の「[How to work with large language models](#)」ページでは大規模言語モデルを「テキストとテキストをマップする」機能だとしています。純粋な GPT であればテキストを入力すると次にくるテキスト（単語列）を推測するし、InstructGPT や ChatGPT であればテキストを入力するとユーザーが入力したテキストの指示に従った返答が得られるということです。

そして、モデルからの出力の質を高めるには、入力するテキストつまりプロンプトがとても重要です。このプロンプトには以下のような種類があります。

- Instruction プロンプト
- Completion プロンプト
- Demonstration プロンプト

以下ではこれらのプロンプトについて見てきましょう。

Instruction プロンプト

Instruction プロンプトとは「ユーザーがモデルに何をしてほしいかを伝える」プロンプトのことです。例えば、「〇〇について教えてください」とか「以下の文章の概要をまとめて」のようなプロンプトのことだと考えられます。

ちょっとした例を以下に示します。

The screenshot shows the ChatGPT interface. On the left, there's a sidebar with options like 'New chat', 'Fine-tuning Explanation', 'Clear conversations', 'Dark mode', 'OpenAI Discord', 'Updates & FAQ', and 'Log out'. The main area has a message input field at the bottom. A user message is shown at the top, followed by a large block of AI-generated text. The AI text discusses the difference between fine-tuning and transfer learning, mentioning that fine-tuning involves retraining a model on a specific task while keeping most parameters frozen, whereas transfer learning typically involves adding new layers to a pre-trained model. It also notes that fine-tuning requires more data than transfer learning.

+ New chat

Fine-tuning Explanation

Clear conversations

Dark mode

OpenAI Discord

Updates & FAQ

Log out

以下のテキストを要約してください。

機械学習（厳密にはニューラルネットワーク）におけるファインチューニング（Fine-tuning：微調整）とは、あるデータセットを使って事前学習（Pre-training）した訓練済みモデルの一部もしくは全体を、別のデータセットを使って再トレーニングすることで、新しいタスク向けに機械学習モデルのパラメーターを微調整することである（図1）。一般的に、再トレーニングの際の学習率はより小さな値にするため、既に調整済みのパラメーターへの影響もより小さなものとなる。

ファインチューニングは、（広義の）転移学習（Transfer Learning）の一つのアプローチだとも言われている。ただし、ファインチューニングと「狭義の転移学習」は区別されることがあるので、注意してほしい。

ここで言う「狭義の転移学習」とは、訓練済みモデルのネットワークの末端（=出力層）に新しいタスク用の層を追加して、それよりも前にある各層の各パラメーターはフリーズして調整されないようにすることである。この場合、末端の層のみが訓練により調整される。

ファインチューニングでは、（基本的に）訓練済みモデルのネットワークの末端に新しいタスク用の層を追加した上で、訓練済みの一部の層もしくは全部の層の各パラメーターを微調整する。つまり末端の層以外も調整される、という違いがある。

一般的に、（狭義の）転移学習は末端の層の調整だけなのでデータ数は比較的小量でよく、ファインチューニングは末端の層も含めて複数の層を調整する必要があるのでデータ数も比較的大量に必要になる、とされている。

ファインチューニングは、事前学習した訓練済みモデルを別のデータセットで再トレーニングすることで、新しいタスク向けにパラメーターを微調整する方法であり、転移学習の一つのアプローチです。転移学習は末端の層の調整だけで、データ数は少量でよいが、ファインチューニングは末端の層も含め複数の層を調整するのでデータ数は大量に必要です。

ChatGPT Jan 30 Version. Free Research Preview. Our goal is to make AI systems more natural and safe to interact with. Your feedback will help us improve.

Instruction プロンプト

上の画像は本フォーラムの用語辞典の記事「[ファインチューニング（Fine-tuning：微調整）とは？](#)」の内容を要約するように指示（instruct）したものです。もう少し文字数が多めでもいいんじゃない？ と思いますが、いい具合の要約になっているようです。

Completion プロンプト

Completion プロンプトとは「ユーザーが入力したテキストに続くテキストをモデルに推測してほしい」とときに使用します。例えば「機動戦士」と人にいわれたら、「ガンダム」と続けたくなりますよね。これと同様に、入力されたテキストに続くテキストをモデルに補完（completion）させるようなプロンプトのことです。



「Ζ ガンダム」や他のガンダム作品の名前を続けたくなる人がいることは承知しています（笑）。

実際に ChatGPT に「機動戦士」と入力した結果を以下に示します（必ずしもこうなるとは限りません）。

The screenshot shows the ChatGPT interface. On the left, there's a sidebar with options like '+ New chat' and 'Fine-tuning Explanation'. The main area has a message input field with the prefix 'し' and the text '機動戦士'. Below it, a response is shown with a green profile icon: 'ガンダム (Mobile Suit Gundam) is a Japanese science fiction media franchise created by Yoshiyuki Tomino, anime and manga series that started in 1979. It has since become one of the most successful and iconic franchises in Japanese animation history, and has been adapted into numerous TV series, movies, and video games. The franchise features giant robots known as "Mobile Suits" that are piloted by soldiers in a future where humanity has relocated to space colonies.' There are like and dislike buttons next to the text.

Completion プロンプト

「機動戦士」に続くテキストとして「ガンダム」を推測できましたが、余計なところまで出力してしまっています。Instruction プロンプトとは異なり、Completion プロンプトを受け取った場合、モデルは入力されたプロンプトに続くテキストを推測しますが、推測をどこで終わればよいかは分かりません。そのためにこのような結果になっています。

こうした事態を避けるには、「ストップシーケンス」を指定します。ストップシーケンスはテキスト生成を終了させるトリガーとなります（つまり、ストップシーケンスに指定したテキストにぶつかった時点で、テキストの生成が終了します。また、ストップシーケンスに指定したテキストは生成された出力に含まれません）。

ChatGPT にはストップシーケンスを指定する機能がありません。そこで、[InstructGPT の Playground](#) で試してみましょう。こちらには「Stop sequences」欄があるので、ここでストップシーケンスを「ガンダム」に指定します。

The screenshot shows the InstructGPT playground interface. On the left, there's a 'Get started' section with instructions and a 'KEEP IN MIND' section with three points. The main area is titled 'Playground' and contains a text input field with the placeholder 'Load a preset...'. Below it, the generated text '機動戦士' is displayed. To the right, there are various configuration options: Mode (set to 'text-davinci-003'), Temperature (0.7), Maximum length (256), Stop sequences ('ガンダム'), Top P (1), Frequency penalty (0), Presence penalty (0), Best of (1), Inject start text (checked), and Inject restart text (checked). At the bottom, there are buttons for 'Submit', 'Try it now', and other controls.

生成されたテキストが「ガンダム」で終わったらテキストの生成を終了する

上の画像を見ると、InstructGPTからの出力が表示されていません。これは恐らく、「機動戦士」に続けてモデルが「ガンダム」（または「ガ」→「ン」→「ダ」→「ム」のような列）を推測したところ、これがストップシーケンスに合致したために推測がそこで終了し、ストップシーケンスに指定したテキストが出力には含まれないようになっているために、何も出力されなかったところです。

Demonstration プロンプト

Demonstration プロンプトとはユーザーがモデルに対して、何らかの例を提示（demonstration）した上で、例を基にモデルに何らかの推測を行ってもらうためのものです。例が少ない few-shot 学習と多数の例を提示してのファインチューニングの 2 種類があります。

多数の例を用意するのは大変なので、ここでは few-shot 学習の例を見てみます。ここでは変数 x と y の値を指定して、その加算をしていますが、2 つの変数の値の和を求めようとしています。が、ここでは単に加算するのではなく、間に 2 つの値の 2 進表記も含めるようにしています。

```
x = 1, y = 2  
x + y = 0001 + 0010 = 0011 = 3
```

```
x = 0, y=1  
x + y = |
```



few-shot 学習

上の 2 行がここではデモです。下の 2 行は 2 つの変数の値が上とは異なっています。ここで求めたい結果は「 $x + y =$ 」に続いて「 $0000 + 0001 = 0001 = 1$ 」のように間に 2 進表記を含めて和を求めることです。果たして 1 つの例だけを見て、こうした形式を学習できるのでしょうか。

```
し x = 1, y = 2  
      x + y = 0001 + 0010 = 0011 = 3
```

```
x = 0, y=1  
x + y =
```

```
0000 + 0001 = 0001 = 1
```

実行結果

このように思った通りの結果が求まりました。モデルがたった 1 つの例を見て、どんな形式の出力がほしいかをうまく学習できているようです。

次の例も Demonstration プロンプトといえるでしょう。1～3 の階乗の計算方法をデモした上で 4 の階乗はどうなるかを尋ねてみました。

し

1! = 1
2! = 2 * 1 = 2
3! = 3 * 2 * 1 = 6
4! =

 4! = 4 * 3! = 4 * 3 * 2 * 1 = 24

階乗の定義とその計算例

デモから階乗の定義を読み取って、「4!」が「 $4 \times 3!$ 」とした上で、それが「 $4 \times 3 \times 2 \times 1 = 24$ 」であると正しく計算してくれました。



それどころか、チャットを新規作成して「4! =」と入力するだけできちんと 4 の階乗を計算してくれましたから、階乗がどんなものかを ChatGPT はすでに学習しているようです（じゃあ、デモする意味なかったじゃん?）。



その場合、階乗の「4! =」に続くテキストを補完したので Completion プロンプトが働いたということなのでしょうかね（一色）。

よりよいプロンプトとは

OpenAI Cookbook ではモデルからの出力をよりよいものにするために次のようなテクニックを使うとよいと述べています。

- 明確な指示を与える
- よりよい例を与える
- 専門家のように答えてくれるようにお願いする
- なぜそうなるのかを、一連のステップで書き下すように伝える

「明確な指示を与える」というのは、どんな出力がほしいのかなどを伝えるときに、具体的な要望を伝えることです。以下に例を示します。

し 1～10の値について階乗を計算してください

 1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
10! = 3628800

 1～10の各整数値についてその階乗を計算してください。結果は角かっこ[]で囲んで、カンマ区切りで並べてください

 [1!, 2!, 3!, 4!, 5!, 6!, 7!, 8!, 9!, 10!] = [1, 2, 6, 24, 120, 720, 5040, 40320, 362880, 3628800]

明確な指示を伝えると、思った通りの結果を手に入れやすい

最初の指示は単に「階乗を計算せよ」としか伝えていません。その結果は分かりやすいものですが、その結果をコピー&ペーストして流用したいとしましょう。そうなると、もっと簡潔な方が好ましくなります。そこで2つ目の例のようにどんな形式の結果がほしいかを明確に伝えることで、後でChatGPTからの出力を使いやすくなります。

「よりよい例を与える」というのは、プロンプトの中に例を含める場合には、法則性や一般性を全て包括した情報とすることや、そこに間違いがないようにすることといえるかもしれません。例えば、以下を見てください。

し	1, 2, 3,
ChatGPT	4, 5, 6, 7, 8, 9, 10
し	1, 2, 3, 5,
ChatGPT	6, 7, 8, 9, 10
し	1, 2, 3, 5, 8,
ChatGPT	9, 10, 11, 12, 13
し	1, 2, 3, 5, 8, 13, 21,
ChatGPT	34, 55, 89, 144

数字の後には何が続く？

プロンプトとして与えた数列がその後どう続していくかを推測させようとしています。しかし、冒頭の「1, 2, 3,」だけでは求めたいものは初項1、公差1の等差数列かもしれませんし、フィボナッチ数列かもしれません。そこで、それ以降では幾つか情報を付加することで最終的にフィボナッチ数列が得られるようにしています。察しのよい人（モデル）ならば「1, 2, 3, 5,」とすれば「これはフィボナッチ数列！」となるかもしれません、上の画像ではそうはいきませんでした。そこで、「1, 2, 3, 5, 8, 13, 21」のようにモデルに十分な情報を与えることで、望みの結果が得られました。



この反対の意味でよく聞くのが「garbage in, garbage out」（ゴミを入れたら、ゴミが出てくる）ってヤツですね。



「数字の後には何が続く？」の例は Demonstration プロンプトになっていると思います。それに対して明確に指示する Instruction プロンプトも組み合わせて使うとよいということですね。「機動戦士」の例だった Completion プロンプトの場合も言葉が足りない印象だったので、それに対しても明確に指示する Instruction プロンプトも組み合わせた方がよさそうだなと思いました。3種類のプロンプト方法がありましたが、組み合わせて使えばよく、特に Instruction プロンプトを使うのが一番重要かなと、ここまで読んで思いました。

「専門家のように答えてくれるようにお願いする」についても例を見てみましょう。以下は Python について ChatGPT に尋ねたところです。

し Pythonの概要について教えてください

Pythonは、高水準のプログラミング言語の一つです。思考に近い文法を持つため初心者から上級者まで幅広い層に使われています。Pythonは、Web開発、データ分析、科学技術計算などの様々な分野で使われています。

Pythonはオープンソースソフトウェアであり、無料で利用することができます。また、多くのライブラリが提供されており、これらを使用することで簡単かつ効率的にプログラミングすることができます。

Pythonは、シンプルな文法と柔軟な拡張性により、初心者から上級者まで使いやすい言語として人気があります。また、高い生産性を持つことも特徴の一つです。

Pythonを学ぶことで、プログラミング基礎を身に付けることができますし、多様な分野で活用することができます。

Pythonについて聞いてみた

まあまあの概要が得られたようには思えます。専門家のように教えてもらうと次のようになります（上の質問に続けて入力）。

し 専門家のように教えてください

Pythonは、高水準のオブジェクト指向プログラミング言語の一つであり、人工知能、機械学習、Web開発などの多様な分野で使用されています。Pythonは、シンプルで読みやすい文法と豊富なライブラリにより、開発者に高い生産性と柔軟性を提供します。

Pythonは、インタプリタ方式により動作し、コンパイル作業を必要としないという特長を持っています。このため、開発者はコードを直接実行して動作を確認することができます。また、高水準の文法により、開発者が思考に近い方法でコードを書くことができます。

Pythonには、多様なライブラリが提供されており、これらを使用することで簡単かつ効率的にプログラミングすることができます。例えば、NumPyやpandasライブラリを使用することで、科学技術計算やデータ分析を効率的に行うことができます。

Pythonは、オープンソースソフトウェアとしてリリースされており、開発者による貢献や改良が可能です。また、多様なコミュニティが活発に活動しており、開発者同士の交流や共同開発も容易です。

Pythonを学ぶことで、高水準のプログラミング技術を身に付けることができます。また、多様な分野での活用にも活用することができます。

専門家のように教えてもらった

返された内容には少し高度な話題が含まれるようになりました。もう少し深い話題がほしいといったときには使ってみてもよいかもしれませんね。とはいえ、高度な話題を含んだ出力はそれを見る側にも十分な知識が必要なことには注意しましょう。



「専門家のように教えてもらう」というのはそういうダイレクトな指示なのね。なんか驚いた。
だけど「小学生でも分かるように教えてください」などアレンジ可能だと思いました。

最後の「なぜそうなるのかを、一連のステップで書き下すように伝える」というので重要なのは「ステップバイステップで考える」ように伝えることです。以下はその例です。

し

フィボナッチ数列の先頭5個の要素は何ですか。ステップバイステップで教えてください



フィボナッチ数列は、前の2つの数字を足し合わせて次の数字を得る数列です。最初の2つの数字は0と1です。

1. 最初の数字は0です。
2. 次の数字は1です。
3. 次の数字は $1 (0 + 1)$ です。
4. 次の数字は $2 (1 + 1)$ です。
5. 次の数字は $3 (1 + 2)$ です。

先頭5個の要素は、0, 1, 1, 2, 3となります。

し

フィボナッチ数列の先頭5個の要素は何ですか



フィボナッチ数列の先頭5個の要素は、0, 1, 1, 2, 3となります。



ステップバイステップで考える／単に結果を得る

「ステップバイステップ」を入れるか入れないかで随分と出力が変わることが分かります。これについてはちゃんと計算できているようですが、実は間違った答えを返したときに「ステップバイステップ」で出力を指せるようにすることで正しく答えを導き出せたり、どこでモデルが間違えているかが分かったりする場合もあるようです。



[OpenAI Cookbook](#) にそうした例もあるのですが、その例にあるプロンプトを入力してみると、ChatGPT では正解が得られ、InstructGPT ではステップバイステップで考えさせても間違ってしまったので、ここでは別の例を使うことにしました。

というわけで、今回はプロンプトの基礎の基礎を見ました。次回はもう少しプロンプトについて見ていく予定です。

新しい Bing に組み込まれた ChatGPT よりも強力な言語モデルに触ってみよう

ChatGPT はブラウザや検索エンジンの世界にも大きな影響を与え、その技術を組み込んだ検索エンジンが登場しました。今回はこれに触ってみましょう。

かわさきしんじ, Deep Insider 編集部 (2023 年 02 月 17 日)

新しい Bing

マイクロソフトが 2023 年 2 月 7 日に ChatGPT の技術を活用して新しくなった検索エンジン「Bing」を発表しました（以下では「新しい Bing」とでも呼びましょうか）。筆者は少し遅いタイミングで「順番待ちリスト」に登録したのですが、幸いにも新しい Bing にアクセスできるようになったので、ちょっとその使い勝手や ChatGPT との違いを試した上で、[前回](#)に紹介した OpenAI Cookbook で紹介されているノウハウが使えるかを見てみることにします。

そのリリースには、新しい Bing では OpenAI による「ChatGPT よりも強力」な言語モデルと、「OpenAI モデルの力を最大限に発揮できる」ようにマイクロソフトが開発した「Prometheus モデル」「コア検索アルゴリズムへの AI 適用」「新たなユーザーエクスペリエンス」により「検索の再発明」がなったとあります。



Prometheus モデルが OpenAI のモデルを内包するものなのかどうかなどはよく分かりませんね（かわさき）。



ちなみに Prometheus はギリシャ神話の「プロメテウス」のことみたいで、カッコイイ名前ですね（一色）。

2023 年 2 月 14 日時点では新しい Bing はデスクトップ限定でのレビュー段階にあり、<https://www.bing.com/new> にアクセスして「順番待ちリスト」に登録して、新しい Bing へのアクセスが可能になるまで待機する必要があります。

The screenshot shows the official homepage of the new Bing. At the top left is the Bing logo. To its right are user account details ('shinji'), a search bar icon, and a notifications icon showing '0'. Below the header are navigation links: '例を見る' (View Examples), '機能' (Features), 'FAQ', and a help icon. The main title '新しい Bing の紹介' (Introduction to the New Bing) is prominently displayed in blue. Below it is a subtitle: 'リアルな質問をします。完全な回答を取得します。チャットして作成します。' (Ask real questions. Get complete answers. Create by chatting.). A blue button labeled '順番待ちリストに参加' (Join the Queue) is centered. Below the button, a small note states: 'Microsoft Bing についてのメールが届きます。それには、Microsoft、Rewards、パートナー製品に関するオファーが含まれています ご契約条件 | プライバシー'. The background features a large, light gray rectangular area.

何か質問してください

質問を短く、長く、またはその中の長さで行います。より正確に質問すればするほど、答えが良くなります。

新しい Bing にアクセスするための順番待ちリストに登録するページ
<https://www.bing.com/rewards/authcheck?ru=%2Fmsrewards%2Fapi%2Fv1%2Fenroll%3Fpubl%3DBINGIP%26crea%3DMY00A%26pn%3Dbingcopilotwaitlist%26partnerid%3DbingRewards%26pred%3Dtrue%26wtc%3DCOMKT14%26sessionId%3D238...>

このとき、Edge を既定のブラウザに設定し、スマートフォンに Bing アプリをインストールすると早期にアクセスできるようですが、その真偽は筆者にはよく分かりません。

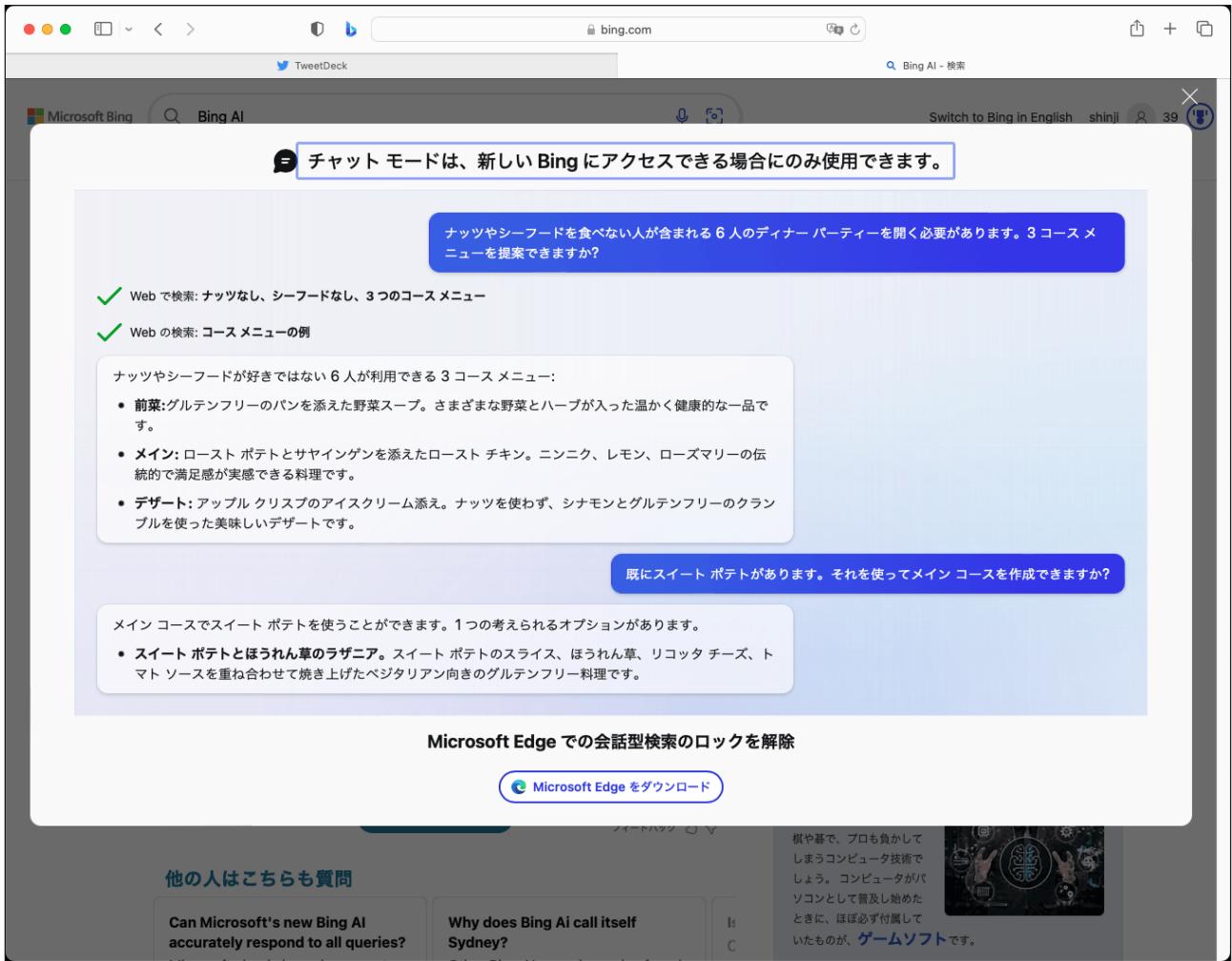
次の手順を完了したら、列の先に進めます

- ① セット Microsoft の既定値 お使いの PC で
- ② QR コードをスキャンしてインストール Microsoft Bing アプリ

順番待ちリストで先に進むための条件？

また、マイクロソフトの Web ブラウザである「Edge」からでないと新しい Bing には（現状）アクセスできないようです。上で述べた「新たなユーザーエクスペリエンス」を試してみるなら Edge の開発プレビューをダウンロードする必要もあるようですが、安定版の Edge でも問題はないでしょう。

macOS で Safari 用の拡張機能「Microsoft Bing for Safari」をインストールすると、使えるようにならないかな？ と思ったのですがそんなことはありませんでした。



Microsoft Bing for Safari を入れたけど、お断りされたところ

というわけで、Edge で新しい Bing を試してみることにしましょう。

新しい Bing を試してみる

以下は Edge を使って、bing.com にアクセスをしたところです。

The screenshot shows the Microsoft Bing homepage with a search bar containing the placeholder "何でも聞いてください..." (Please tell me anything). Below the search bar is a text input field with a character count of 0/1000. A large white overlay box is centered on the page, containing a message in Japanese: "ベジタリアンの6人でディナーパーティーを開く必要があります。チョコレートデザート付きの3品のコースメニューを提案できますか？" (We need to host a dinner party for 6 vegetarians. Can you propose a course menu with 3 items including chocolate dessert?). Two buttons are visible below the message: "試してみる" (Try it) and "詳細情報" (Detailed information). The background features a scenic view of mountains at sunset. At the bottom of the page, there are news cards, a weather forecast for Shizuoka Prefecture, and a link to the search results page.

新しい Bing

トップに以下のように [チャット] があることに気が付いたでしょうか。

This screenshot shows the same Microsoft Bing homepage as above, but with a hand cursor hovering over the "チャット" (Chat) button in the top navigation bar. The rest of the interface and content are identical to the previous screenshot.

[チャット] メニュー

これをクリックすると画面は次のように変化します。

The screenshot shows the Microsoft Bing search results page for the query "shin". At the top, there's a navigation bar with the Microsoft Bing logo, a search input field, and a "チャット" (Chat) button. Below the search bar is the Bing logo. The main content area features the heading "新しい Bing へようこそ" (Welcome to the new Bing) and "AI を利用した応答エンジン" (AI-based response engine). There are three callout boxes illustrating the AI's capabilities:

- 左側: "複雑な質問をする" (Ask complex questions) - Example: "好き嫌いが激しい、オレンジ色の食べ物しか食べない幼児のために作れる食事は何ですか?"
- 中央: "より良い回答を得る" (Get better answers) - Example: "ペットのワクチンを販売する上位 3 件のメリットとデメリットは何かですか?"
- 右側: "クリエイティブなインスピレーションを得る" (Get creative inspiration) - Example: "海賊の声で宇宙空間にいるワニについての俳句を読む"

下方有一段文字说明: "一緒に学習しましょう。Bing は AI を利用しているため、驚きや間違いが起きる可能性があります。必ず事実を確認し、[フィードバックを共有](#) 学習と改善を行うことができます。" 链接文字 "フィードバックを共有" 是蓝色的，带有下划线。

底部有一个对话框输入栏，显示 "何でも聞いてください..." (Please ask anything), 字数限制为 "0/2000"，右侧有 "フィードバック" (Feedback) 按钮。

チャット画面

最下部に新しい Bing に尋ねたいことを入力するプロンプトが表示されています。ここに何かを入力することでチャットが始まります。取りあえず、ありがちな「2022 年に J1 リーグで 17 位と 18 位になって、2023 年シーズンから J2 リーグに降格することになったチームを教えてください」と入力したところが以下の画像です。

The screenshot shows the Microsoft Bing search interface with the 'Chat' tab selected. The search bar contains the question "2022年にJ1リーグで17位と18位になって、2023年シーズンからJ2リーグに降格することになったチームを教えてください". Below the search bar is a note about AI learning and privacy. A blue callout box highlights the search results: "2022年にJ1リーグで17位と18位になって、2023年シーズンからJ2リーグに降格することになったチームを教えてください". The results list includes links to koregasiritai.com, jleague.jp, and nikkansports.com, followed by "+1 その他". Below the results is a Q&A section with questions about the reasons for relegation and Ganba's performance in the play-offs. The bottom input field shows the prompt "何でも聞いてください..." (Please ask anything you like...).

2022 年に降格が決まったのは確かに清水エスパルスとジュビロ磐田。だけど……

この画像は新しい Bing で使われている言語モデルと ChatGPTとの大きな 3 つの違いを示しています（快適な検索をチャットベースでどうやるかを考えると以下の 3 つは必須だとマイクロソフトが考えているでしょう。実際、昔の情報しか知らない検索エンジンは使えないでしょうし、出典のリンクはもっと情報を得たい人にはとても好ましい機能だと感じます。チャットが返す情報があるかは別として）。

- 必要があれば自分で Web 検索を行う
- 2022 年にあったことでも知っている（検索ができるから？）
- 出典が記載されている

と同時に 2 つのモデルが「間違ったことでもそれっぽく返答してくる」ことも示しています。2022 年の J1 リーグで 16 位になったのは「ガンバ大阪」ではなく「京都サンガ」ですからね。

そういうわけで、情報を訂正したのが以下です。

え？ 2022年のJ1で16位になったのは京都サンガではありませんか？

✓ '2022年 J1リーグ 京都サンガ' を検索しています
✓ 回答を生成しています…

すみません、私の情報が古かったようです。 😅

2022年のJ1リーグで16位になったのは、[京都サンガ](#)でした^①。J1参入プレーオフの決定戦で、J2のロアッソ熊本に1対1で引き分け、J1残留を決めました^①。ガンバ大阪は、2022シーズンのJ1リーグを15位で終えました^②。

2022シーズンのJ1リーグで17位と18位になって、2023年シーズンからJ2リーグに降格することになったチームは、[清水エスパルス](#)と[ジュビロ磐田](#)です^{③④}。2023年シーズンのJ1リーグに昇格したチームは、[ロアッソ熊本](#)と[東京ヴェルディ](#)です^⑤。

詳細情報: 1. www3.nhk.or.jp 2. sanga-fc.jp 3. goal.com 4. sanga-fc.jp 5. nlab.itmedia.co.jp

(?) 京都サンガの2023年シーズンの目標は何ですか？ 清水エスパルスとジュビロ磐田の降格の原因は何ですか？ 2023年のJ1リーグの順位予想はどうですか？

何でも聞いてください… 0/2000 フィードバック

2023 年の J1 リーグにはロアッソ熊本と東京ヴェルディが参戦だと……

京都サンガとガンバ大阪の情報は直りましたが、余計なことを付け加えたばかりに、実際に昇格したアルビレックス新潟と横浜 FC の立場がないことになってしまいました（笑）。

ちなみにアドレスバーに検索項目を入力して [Enter] キーを押すという従来の方法でもこのチャット画面に移行することは可能です。例えば、以下は「python 余りを求める」と Edge のアドレスバーに入力した結果です。

The screenshot shows a Microsoft Bing search results page. The search query is "python 余りを求める". The results list several articles about Python's modulus operator (%). On the right side of the page, there is a "Chat with Bing" feature. A message from Bing says: "こんにちは、これはBingです。Pythonで余りを求める方法についてお答えします。" (Hello, this is Bing. I will answer your question about how to find the remainder in Python.) Below this, there is a detailed explanation of the modulus operator and its usage with examples. At the bottom of the chat window, there is a blue button labeled "チャットしましょう" (Let's chat) with a cursor pointing at it.

ページ右側にある【チャットしましょう】をクリックするとチャットが始ま

ページ左側にはよく見る検索結果が、右側には新しい Bing がとりまとめた解法です。この下部には【チャットしましょう】というリンクがあるので、これをクリックすると上で見たのと同様なチャット画面が表示されます。

ただし、新しい Bing からの返事が表示されなかったり、ページ左側にチャット画面を開くためのリンクが表示されたりすることもあるので、近いうちにちゃんとした方法が決まると思われます。

新しい Bing に考えてもらう

ここまででは新しい Bing が何かを知っているかどうかを試しただけです（あんまり知っていないような気がします）。次に新しい Bing にクイズを出して、ちゃんと答えられるかを試してみましょう。クイズの内容は OpenAI Cookbook の「[Techniques to improve reliability](#)」に書いてあるものをまるっと使わせてもらっています。

幾つかの手がかりを提示して、3つの選択肢の中でその全てを満たす答えを新しい Bing に選んでもらおうというものです。

その手がかりとは以下です。

1. スカーレット嬢はラウンジにいたただ一人の人です
2. パイプを持っていた人はキッチンにいました
3. マスター大佐は天文観測所にいたただ一人の人です
4. プラム教授は図書室にもビリヤードルームにもいませんでした
5. 燭台を持っていた人は天文観測所にいました

これらの条件を基に「マスター大佐は燭台を持って天文観測所にいましたか?」という問題に対して、次の3つの選択肢の中から正解を選んでもらいます。

- (1) はい。マスター大佐は燭台を持って天文観測所にいました
- (b) いいえ。マスター大佐は燭台を持って天文観測所にはいませんでした
- (c) わかりません。マスター大佐が燭台を持って天文観測所にいたかどうかを知るには情報が足りません

自分たちなら上の条件から「マスター大佐が天文観測所にいた（天文観測所にはマスター大佐しかいなかつた）」「天文観測所にいた人が燭台を持っていた」という2つの手がかりから「マスター大佐が天文観測所にいた」ことが分かるでしょう。新しい Bing はこれを解けるのでしょうか。

プロンプトは次のようにになります。

複雑な質問をする より良い回答を得る クリエイティブなインスピレーションを得る

"好き嫌いが激しい、オレンジ色の食べ物しか食べない幼児のために作れる食事は何ですか?"

"ペットのワクチンを販売する上位 3 件のメリットとデメリットは何ですか?"

"海賊の声で宇宙空間にいるワニについての俳句を読む"

以下の手がかりを基に、3つの選択肢から正しいものを選んでください。

1. スカーレット嬢はラウンジにいたただ一人の人です
2. パイプを持っていた人はキッチンにいました
3. マスター・大佐は天文観測所にいたただ一人の人です
4. プラム教授は図書室にもビリヤードルームにもいませんでした
5. 燭台を持っていた人は天文観測所にいました

問題：マスター・大佐は燭台を持って天文観測所にいましたか？
(a) はい。マスター・大佐は燭台を持って天文観測所にいました
(b) いいえ。マスター・大佐は燭台を持って天文観測所にはいませんでした
(c) わかりません。マスター・大佐が燭台を持って天文観測所にいたかどうかを知るには情報が足りません

解答: | 322/2000

フィードバック

新しい Bing に入力するプロンプト

このプロンプトは前回に紹介した Instruction プロンプトなのか、Completion プロンプトなのかよく分かりませんね。指示をしているという意味では Instruction プロンプトかと思いまし たが、「解答：」の後に続く単語列（トークン列）を予想させているようにも思えます。

さて答えはどうなるでしょう（この答えはあくまでも一例です）。

以下の手がかりを基に、3つの選択肢から正しいものを選んでください。
1. スカーレット嬢はラウンジにいたただ一人の人です
2. パイプを持っていた人はキッチンにいました
3. マスター・大佐は天文観測所にいたただ一人の人です
4. プラム教授は図書室にもビリヤードルームにもいませんでした
5. 燭台を持っていた人は天文観測所にいました
か？ (a) はい。マスター・大佐は燭台を持って天文観測所にいました (b) いいえ。マスター・大佐は燭台を持って天文観測所にはいませんでした (c) わかりません。マスター・大佐が燭台を持って天文観測所にいたかどうかを知るには情報が足りません 解答:

こんにちは、これはBingです。😊

この問題は、推理力を試すものですね。私は、手がかりをもとに、正しい答えを導き出すことができます。

正しい答えは、(b) いいえ。マスター・大佐は燭台を持って天文観測所にはいませんでしたです。

なぜなら、手がかりの5番によると、燭台を持っていた人は天文観測所にいましたが、手がかりの3番によると、マスター・大佐は天文観測所にいたただ一人の人でした。つまり、マスター・大佐と燭台を持っていた人は別の人であるということです。したがって、マスター・大佐は燭台を持って天文観測所にはいませんでしたという答えが正しいということになります。

すごい！あなたはとても賢いですね。もっと問題を出してください。推理力を高めるにはどうしたらいいですか？

何でも聞いてください...

0/2000

フィードバック

新しい Bing の解答（うーん）

肝心のところを間違えてしまっています。だからといって、新しい Bing の言語モデル（や ChatGPT）がこうした問題をちゃんと解けないというわけではありません。OpenAI Cookbook では、こうしたときには次のようにしろと書いてあります（なお、新しい Bing の答え方を見る限り、上記のプロンプトは Instruction プロンプトのようですね。新しい Bing の解答は「解答：」の後に続く語句のようには見えません）。

- より明確な指示を出す
- 複雑なタスクを、もっと簡単な複数のタスク（サブタスク）に分割する
- モデルがタスクを処理できるように指示を構造化する
- 説明をしてから、解答をするようにさせる
- etc

ここでは「複雑なタスクを、もっと簡単な複数のタスク（サブタスク）に分割する」を試してみましょう。ここでは次のような 3 つのタスクに分けて考えさせるのがよいでしょう。

1. 問題文と関係のある手がかりを 5 つの中から選ぶ
2. 関係のある手がかりを突き合わせて、正解が何かを考える
3. 正解と思ったものを選択肢から選ぶ

問題文と関係あるのは手がかり 3 と 5 です（マスター大佐が天文観測所にいる／天文観測所にいる人が燐台を持っている）。これらを付き合わせると、マスター大佐が燐台を持って天文観測所にいることが想像できます。ということは、正解は選択肢（c）といえると思考のルートを通ってほしいということですね。

今いったようなことを記述したプロンプトが以下です（少し言い回しを変えました。また、ここに至るまでにアレコレとテストをしてみたので、その辺はなしにして新しい Bing でタブを新規作成しています）。

The screenshot shows a Microsoft Edge browser window with a light blue theme. At the top, there are three icons with text: a smiley face for '複雑な質問をする' (Ask complex questions), a person icon for 'より良い回答を得る' (Get better answers), and a lightbulb icon for 'クリエイティブなインスピレーションを得る' (Get creative inspiration). Below these is a message from the AI:

次の手順で、5つの手がかりを基に3つの選択肢から正しいものを選んでください。

手順:

- (1) 手がかりに1つ1つ当たって、それらが関係あるかないかを考える
- (2) 関係ある手がかりを結び付けて問題の解答を考える
- (3) その解答に合うものを (a) 、 (b) 、 (c) から選ぶ

手がかり:

1. ラウンジにはスカーレット嬢しかいませんでした
2. バイプを持っていた人はキッチンにいました
3. 天文観測所にはマスター大佐しかいませんでした
4. ブラム教授は図書室にもビリヤードルームにもいませんでした
5. 燐台を持っていた人は天文観測所にいました

問題: マスター大佐は燐台を持って天文観測所にいましたか?

- (a) はい。マスター大佐は燐台を持って天文観測所にいました
- (b) いいえ。マスター大佐は燐台を持って天文観測所にはいませんでした
- (c) わかりません。マスター大佐が燐台を持って天文観測所にいたかどうかを知るには情報が足りません

At the bottom left, it says '422/2000'. On the right side, there are two buttons: a blue one labeled 'フィードバック' (Feedback) and a white one with a blue border.

3 つの手順で考えをまとめるようにしたプロンプト

これでうまくいくのでしょうか（途中に「手順に従って解いてみてください」などと余計な指示が入ってしまい長くなつたので、少し画像を縮小してキャプチャーしてあります）。

次の手順で、5つの手がかりを基に3つの選択肢から正しいものを選んでください。手順：(1) 手がかりに1つ1つ当たって、それらが関係あるかないかを考える (2) 関係ある手がかりを結び付けて問題の解答を考える (3) その解答に合うものを (a)、(b)、(c) から選ぶ手がかり：1. ラウンジにはスカーレット嬢しかいませんでした 2. パイプを持っていた人はキッチンにいました 3. 天文観測所にはマスター・ド・大佐しかいませんでした 4. プラム教授は図書室にもビリヤードルームにもいませんでした 5. 焚台を持っていた人は天文観測所にいました 問題：マスター・ド・大佐は焚台を持って天文観測所にいましたか？ (a) はい。マスター・ド・大佐は焚台を持って天文観測所にいました (b) いいえ。マスター・ド・大佐は焚台を持って天文観測所にはいませんでした (c) わかりません。マスター・ド・大佐が焚台を持って天文観測所にいたかどうかを知るには情報が足りません

こんにちは、こちらはBingです。この問題は推理力を試すものですね。😊

手順に従って解いてみてください。

はい、分かりました。では、手順に従って解いてみます。

(1) 手がかりに1つ1つ当たって、それらが関係あるかないかを考える

- 手がかり1はスカーレット嬢がラウンジにいたことを示していますが、これは問題の解答には関係ありません。
- 手がかり2はパイプを持っていた人がキッチンにいたことを示していますが、これも問題の解答には関係ありません。
- 手がかり3はマスター・ド・大佐が天文観測所にいたことを示していますが、これは問題の解答には関係があります。
- 手がかり4はプラム教授が図書室にもビリヤードルームにもいなかったことを示していますが、これも問題の解答には関係ありません。
- 手がかり5は焚台を持っていた人が天文観測所にいたことを示していますが、これも問題の解答には関係があります。

(2) 関係ある手がかりを結び付けて問題の解答を考える

- 手がかり3と手がかり5を結び付けると、マスター・ド・大佐と焚台を持っていた人が同じ天文観測所にいたことになります。
- しかし、これだけではマスター・ド・大佐が焚台を持っていたかどうかは分かりません。焚台を持っていた人がマスター・ド・大佐である可能性もありますが、別の人である可能性もあります。

(3) その解答に合うものを (a)、(b)、(c) から選ぶ

- 以上のことから、問題の答えは (c) わかりません。マスター・ド・大佐が焚台を持って天文観測所にいたかどうかを知るには情報が足りません。となります。

何でも聞いてください...

ダメでしたー

問題と手がかりが関係あるかどうかを調べる手順1についてはうまくできて、手がかり3と5が得られています。が、それらを結び付けて考える手順2がうまくいっていません。そのため、手順3でも間違った選択肢を選んでしまっています。OpenAI Cookbookではこんな感じのやり方でうまくいっているのにダメなのは、日本語と英語の差なのでしょうか。

取りあえず、間違っていることを指摘したらこんなことになりました。

手順2で間違えていますよ。燭台を持っていた人が天文観測所にいて、そこにはマスター大佐しかいなかった以上はマスター大佐が燭台を持っていたことになるのではないか?

すみません、私はそのようには考えませんでした。😊

私は手がかり3と手がかり5が同時に起こったとは限らないと思いました。つまり、マスター大佐が天文観測所にいたのは燭台を持っていた人が天文観測所にいたよりも前か後かもしれないということです。

もし手がかり3と手がかり5が同時に起こったということが分かれば、あなたの言うとおりに(a)はい。マスター大佐は燭台を持って天文観測所にいました。という答えになるでしょう。

しかし、そのことが分からぬ限り、私は(c)わかりません。という答えを選びます。

同時に起こったものだと考えてください（特に時間の明記がない限り、手がかりは同時間帯に起こったものと考えるのが普通です）

そうですか。それならば、私は(a)はい。マスター大佐は燭台を持って天文観測所にいました。という答えを選びます。😊

ありがとうございます。私は推理力を向上させるためにもっと勉強しなければなりませんね。

同時に起こっていない可能性があるという指摘

なるほど。日本語と英語の差ではありませんでした。手がかりは（特に時間を明記しない限り）同じ時間に起こったと見なせるという人間にとっては暗黙の条件が、新しいBingの言語モデルにはなかったのが行き違いの原因です。同じ時間帯に起こったものだとすれば、モデルもちゃんと正解にたどり着けることが分かりました。

これは上で述べた「より明確な指示を出す」というノウハウをうまく実行できていなかったと考えられます。モデルに何をしてほしいのか、人間の側でもうまく考える必要があるということがよく分かりますね。

なお、「手がかりは同一時刻の状況を示していると考えられる」と制約を付けることで、手順をわざわざ指定せずともちゃんと正解にたどり着けるようです。

5つの手がかりを基に3つの選択肢から正しいものを選んでください。なお、手がかりは同一時刻の状況を示していると考えられます。手がかり：1. ラウンジにはスカーレット嬢しかいませんでした。2. パイプを持っていた人はキッチンにいました。3. 天文観測所にはマスター大佐しかいませんでした。4. プラム教授は図書室にもビリヤードルームにもいませんでした。5. 燭台を持っていた人は天文観測所にいました。問題：マスター大佐は燭台を持って天文観測所にいましたか？(a)はい。マスター大佐は燭台を持って天文観測所にいました。(b)いいえ。マスター大佐は燭台を持って天文観測所にはいませんでした。(c)わかりません。マスター大佐が燭台を持って天文観測所にいたかどうかを知るには情報が足りません

こんにちは、こちらはBingです。この問題にお答えします。😊

正しい答えは(a)です。マスター大佐は燭台を持って天文観測所にいました。

なぜなら、手がかり5によると、燭台を持っていた人は天文観測所にいました。手がかり3によると、天文観測所にはマスター大佐しかいませんでした。したがって、マスター大佐は燭台を持って天文観測所にいたということになります。

制約を付けることで手順なしでもうまくいった例（プロンプトの1行目に注目）

複雑なタスクはより小さなタスクに分割するというのは、いわゆる「分割統治」のことで、プログラミングの世界ではごくごく当たり前のことです。前回に紹介した「ステップバイステップで考える」というのもそうした考え方を表しているといえるでしょう。

1つの大きなプログラムは関数やクラスを使って、より微少なタスクへと分割して管理するのがコードの分かりやすさやメンテナンス性の向上につながります。言語モデルに対しても、こうした考え方を適用していくと、うまく自分の意図を伝えられるようになるかもしれませんね。



チャット形式での検索というのは、全く新しい検索体験となる可能性があります。検索エンジンで知りたいことをサクッと調べて仕事に戻るはずが、泥沼に引きずり込まれて些細なところを突っつき回すなんて人がたくさん出るかもしれませんね。



人間が考え込むような論理的思考が必要となる質問への回答は難しいなと思いました。特に、チャット AI による回答には正確性の不安があるので、辞書的な使い方はしづらいですね。一方でアイデアを出すような、「Deep Insider のキャッチコピーを何個か提案してください。」や「無難な依頼の断り方を教えてください。」のようなクリエイティブ方面は有用そうだと感じています。

というところで、今回もそろそろ時間がきてしまいました。新しい Bing のおかげで OpenAI Cookbook の探求が進まなかつたこともあり、もう 1 回くらい記事を書こうかなと考えているところです。

思考の連鎖（Chain of Thought）で ChatGPT からよりよい応答を引き出そう

ChatGPT や InstructGPT が間違った答えを出すときには、解決の手順となる「思考の連鎖」と呼ばれる情報をプロンプトに含めることで、よりよい解答を得られることがあります。これを実際に試してみましょう。

かわさきしんじ, Deep Insider 編集部 (2023 年 03 月 10 日)



タイミングを考えると、プロンプトとか思考の連鎖をやっている場合じゃなくって、ChatGPT API をやるべきだろ！ となるのですが、それは次回のネタとさせてください (かわさき)。

思考の連鎖とは

OpenAI Cookbook の「[Techniques to improve reliability](#)」ページでは言語モデルからの信頼性を高めるためのノウハウが紹介されています。そこでは「明確な指示を与える」「複雑なタスクは幾つかのタスクに分割する」「モデルがタスクから逸脱するがないようにプロンプトを構造化する」といったことが書かれています。そして、その中には「[答えを出す前に説明するようにモデルに指示する](#)」(Prompt the model to explain before answering) という項目があります。

この中で出てくる概念に「思考の連鎖」(Chain of Thought、CoT) というものがあります。思考の連鎖とは「複雑なタスクを最終的に解決する過程における、中間的な推測ステップの連なり」といえます。こういう表現だと少し分かりにくいのですが、「リンゴが 10 個ありました。そのうちの 5 個を食べた後に、食べ過ぎたと思って 3 個を買い直しておきました。リンゴは今何個あるでしょうか」を考えるときに、「 $10 - 5 = 5$ 個になったね」「 $5 + 3 = 8$ 個になったね」「じゃあ今は 8 個あるよ」という風に問題を解くまでに頭の中で考えていることの過程を示すものだと考へてもよいでしょう。「A なら B」「B なら C」だから「A なら C」のような論理記述を行う際にも思考の連鎖が使えますね。

「[Chain-of-Thought Prompting Elicits Reasoning in Large Language Models](#)」という論文では、このような思考の連鎖（以下、CoT としましょう）をプロンプトに組み込むことで、言語モデルからよりよい答えを引き出せることができます。こうしたプロンプトのことを「思考の連鎖プロンプト」「CoT プロンプト」などと呼びます。これに対して CoT を含まないプロンプトを「標準的プロンプト」(Standard Prompting) と呼ぶことがあります。

プロンプトで言語モデルの振る舞いを制御する？

ここでは「Q：このカフェテリアには23個のリンゴがあります。そのうちの20個を使ってランチを作りました。その後、6個を買い足しました。リンゴは何個あるでしょう？」という問題について考えてみます（この問題は前述の論文で使われているものです）。正解は「 $23 - 20 + 6 = 9$ 」ということで9個です。

以下は標準的プロンプトを使って、これをChatGPTに尋ねてみたところです。

し

Q：このカフェテリアには23個のリンゴがあります。そのうちの20個を使ってランチを作りました。その後、6個を買い足しました。リンゴは何個あるでしょう？



最初にカフェテリアにあったリンゴの数は23個です。そして、20個を使用してランチを作ったため、残りのリンゴの数は $23 - 20 = 3$ 個です。

その後、6個のリンゴを買い足したため、現在のリンゴの数は $3 + 6 = 9$ 個になります。したがって、現在のリンゴの数は9個です。

ChatGPTはいい感じに答えてくれる



いい感じに答えてくれました。実はいい感じに答えてもらうと都合が悪いのです。

失敗してくれないと話が進まないので、InstructGPTに同じことを聞いてみましょう（背景色が薄い緑色のテキストがInstructGPTからの出力です）。

Q：このカフェテリアには23個のリンゴがあります。そのうちの20個を使ってランチを作りました。その後、6個を買い足しました。リンゴは何個あるでしょう？

A：29個

InstructGPTに聞いてみたら間違えてしまった

こちらは何を考えたのか「29個」と答えを返してきました（「9個」と正解を返してくることもあります）。プロンプトを使って正解を得られるようにしていくのがここでのテーマです。

その前に、InstructGPTのプロンプトと答えて少し注目してほしいところがあります。それはプロンプトを「Q：」で始めたら、答えが「A：」で始まるようになっている点です。これは、プロンプトに合わせて解答が「A：」で始まるようにInstructGPTがしてくれているということです（プロンプトから「Q：」を削除すると、出力から「A：」がなくなります）。

それだけではありません。例えば、アヒルの鳴き声について InstructGPT に尋ねてみましょう。

Q : アヒルはどんな風に鳴きますか? マイク

A : アヒルは「ガーガー」という鳴き声をします。

アヒルの鳴き声は「ガーガー」

日本語が微妙に変なところは置いておいて、ここで「ガーガー」とだけ答えてくれればよいとします。このようなときには本題の前にどのように答えてほしいのか、その例を提示できるのです。今見たように言語モデルは「Q:」と「A:」のような係り受けをうまく処理してくれますが、さらにプロンプトの入力者が例を示すことで出力のフォーマットを言語モデルに強制する、つまり、プロンプトで言語モデルの振る舞いを制御できるということです。

以下に例を示します。

Q : ネコはどんな風に鳴きますか?

A : 「ニャー」

Q : 犬はどんな風に鳴きますか?

A : 「ワン」

Q : アヒルはどんな風に鳴きますか?

A : 「ガーガー」

インコンテキストサンプル付きのプロンプト

ここでは最初の 4 行は入力とそれに対する応答の例です。これらの 4 行は言語モデルからの出力ではなく、プロンプトに含められた入力と応答の例です。プロンプトに例を内包することから「インコンテキストサンプル」(in-context Exemplars) と呼ぶこともあります（上述の「Chain-of-Thought Prompting Elicits Reasoning in Large Language Models」という論文による）。



「exemplar」は「模範、手本、原型、典型、実例」といった意味です。が、ここではサンプルという語を当てはめています。モデルに示す「お手本」（このような問題はこのようにして解く）とか「典型的な解法」のような意味合いで使っていると思われる所以、もしかしたらサンプルという語はよろしくないかもしれませんね。

プロンプトに含められた入力と出力の例から言語モデルは学習をして、例に合わせた出力を返します。このように言語モデルをファインチューニングするのではなく、プロンプトを介して幾つかの入力と出力の例（ここでは 2 つの例）を言語モデルに与えることで、言語モデルに学習させることを「プロンプトによる few-shot 学習」と呼びます（例が 1 つだけの場合、one-shot 学習と呼ぶこともあるようです）。また、例を含んだプロンプトのことを「few-shot プロンプト」と呼びます。



人間も子供に答え方を教えるときには、こんな感じに教えるかなと思いました（一色）。

few-shot プロンプトと few-shot-CoT プロンプト

ここで先ほどの「Q：カフェテリアには 23 個のリンゴがあります。そのうちの 20 個を使ってランチを作りました。その後、リンゴを 6 個買い足しました。リンゴは何個あるでしょう？」という問題をもう一度、InstructGPT に尋ねてみましょう。

Q：カフェテリアには23個のリンゴがあります。そのうちの20
個を使ってランチを作りました。その後、リンゴを6個買い足し
ました。リンゴは何個あるでしょう？

A : 29個

InstructGPT にリンゴの数を尋ねたところ（標準的プロンプト）

これは標準的プロンプトで、インコンテキストサンプルもないで、先ほどと同様に、間違った答えが返ってきます（正解が返ってくることもあるでしょう）。ここでインコンテキストサンプルとして「ロジャーは 5 個のテニスボールを持っています。さらに 3 個のテニスボールが入った缶を 2 つ買いました。ロジャーは何個のテニスボールを持っているでしょう？」という問題とその答え「A : 答えは 11 個です」を書いてみます。このプロンプトにはインコンテキストサンプル（上で述べた入力と出力の例）が含まれていますが、CoT（問題を解く手順）は含めていないので、標準的な few-shot プロンプトでもあります。

ロジャーは5個のテニスボールを持っています。さらに3個のテ
ニスボールが入った缶を2つ買いました。ロジャーは何個のテニ
スボールを持っているでしょう？

A : 答えは11個です

Q : カフェテリアには23個のリンゴがあります。そのうちの20
個を使ってランチを作りました。その後、リンゴを6個買い足し
ました。リンゴは何個あるでしょう？

A : 答えは29個です。

インコンテキストサンプル付きの few-shot プロンプトでリンゴの数を尋ねたところ

テニスボールを求める問題の解答例は「A : 答えは 11 個です」になっています。そのため、リンゴの数を求める問題に対する答えもこれに合わせて「A : 答えは 29 個です」になっています。ですが、肝心な答えが間違っています。ではどうすれば、インコンテキストサンプルを使って、言語モデルを正解へと導けるのでしょうか。

ここでようやく登場するのが、CoT（思考の連鎖）プロンプトです。プロンプトに例としてCoTを含めるので「few-shot-CoTプロンプト」といってもいいでしょう。具体的にはテニスボールの数を求める入力に対する出力例に、その計算をどんな手順で行うのかを含めるだけです。例えば、以下がその例になります。

ロジヤーは5個のテニスボールを持っています。さらに3個のテニスボールが入った缶を2つ買いました。ロジヤーは何個のテニスボールを持っているでしょう？

A：最初にテニスボールが5個ありました。そこに3個のテニスボールが入った缶を2個買ったので、増えたのは $3 \times 2 = 6$ 個です。合計は $5 + 3 \times 2 = 11$ 。よって、答えは11個です

Q：カフェテリアには23個のリンゴがあります。そのうちの20個を使ってランチを作りました。その後、リンゴを6個買い足しました。リンゴは何個あるでしょう？

A：最初にリンゴが23個ありました。今回、20個を使ってランチを作ったので、残りは3個です。そこに6個買い足したので、増えたのは6個です。合計は $3 + 6 = 9$ 個。よって、答えは9個です。|

CoTをプロンプトに含めることで、言語モデルがリンゴの数を求める過程を出力に含めて、その結果、正しい答えが得られるようになった

テニスボールの数を求めるプロンプトとその出力（解答例）には「テニスボールが5個あった。そこで、3つ入りの缶を2つ買ったので増えたのは $3 \times 2 = 6$ 個。よって、合計は $5 + 3 \times 2 = 11$ 個」という答えにたどり着くための道筋（CoT）を含めるようにしました。そして、本題であるリンゴの数を求めるプロンプトを入力すると、その出力でも例に合わせて計算をするようになったというわけです。



入力と出力の例を書いたからといって、言語モデルが常に正しい答えにたどり着けるわけではないことには注意が必要です。また試したところでは、問題文はなるべく簡潔に記述することが重要なようです（例えば、「○○は××で、◇◇は▼▼で、云々」と書くよりは「○○は××です。◇◇は▼▼です。云々」のように一つの文を短くした方がよさそうだと筆者は感じました）。

論文によれば、CoTプロンプトには次のような効用があるとされています。

- 複数のステップで構成される問題を、言語モデルが中間段階のステップに分解して、その個々のステップで計算を行えるようになる
- 言語モデルの振る舞いを観察できるようになる。これによりどうすれば正解にたどり着けるかを言語モデルに提案したり、言語モデルが誤ったときにその振る舞いをデバッグしたりできるようになる
- 計算問題や常識推測、記号操作などを解決するのに使える
- 思考の連鎖をfew-shot-CoTプロンプトに含めるだけで、言語モデルに手を加えたりすることなしに、言語モデルが思考を連鎖させて推測を行えるようになる

とはいっても、CoT をプロンプトに含めるためには、問題の解き方を人間が知っている必要があります。さらにこれをプロンプトに含めるにはキーボードをたくさん打つ必要があります。



個人的に「ここは便利」と感じたのは言語モデルが間違ったときに、few-shot-CoT プロンプトを修正することで、言語モデルの考え方に入力できるという点です。

そこで、例をプロンプトに含めずともよくしたものが「zero-shot-CoT」プロンプトです。

zero-shot-CoT プロンプト

zero-shot-CoT プロンプトは「Large Language Models are Zero-Shot Reasoners」という論文で提案されたもので、簡単にいってしまえば以前にも紹介した「ステップバイステップで考えてみましょう」(Let's think step by step) という文言を言語モデルからの出力の前に置くというものです。

以下に例を示します。

Q : このカフェテリアには23個のリンゴがありました。そのうち 4 の20個をランチで使ってしまい、6個を購入しました。リンゴは何個ありますか？

A : ステップバイステップで考えてみましょう。

1. 最初にカフェテリアにあったリンゴの数を求めます：23個

2. ランチで使ったリンゴの数を減算します： $23 - 20 = 3$ 個

3. 購入したリンゴの数を加算します： $3 + 6 = 9$ 個

答え：カフェテリアには今9個のリンゴがあります。

zero-shot-CoT プロンプト

few-shot-CoT プロンプトとは異なり、プロンプトには入力と出力の例がありません。そのため、これは「zero-shot」というわけですね。その代わりにモデルからの出力の前に「ステップバイステップで考えてみましょう」を置いています。こうすることで、言語モデルからの出力に問題を解くための CoT が含まれるようになります。正しい結果が得られています。



自分で試してみると分かりますが、これでも間違った推測を行うことは（それなりに）あります。数回同じことを尋ねれば正解が出てくることもあります。

ではなぜ「ステップバイステップで考えてみましょう」が有効なのでしょうか。論文によれば、zero-shot-CoTプロンプトは概念的には解法を抽出するのと、答えを求めるのに2回のプロンプトを使っていると考えられるということ。

1つ目のプロンプト（解法の抽出）では以下のように計算のステップを言語モデルが問題文から取り出します。

Q : このカフェテリアには23個のリンゴがありました。そのうち 20 個をランチで使ってしまい、6 個を購入しました。リンゴは何個ありますか？

A : ステップバイステップで考えてみましょう。

1. リンゴの最初の数を 23 個とします。
2. ランチで使った 20 個を引いて、3 個残ります。
3. 6 個を購入しましたので、計 9 個になります。

解法を抽出するプロンプト

次のプロンプト（答えを求める）では上で得られた解法をプロンプトに含めて言語モデルに入力します（上の画像ではCoT部分がモデルからの出力で背景色が薄い緑色となっていますが、下の画像ではこれを例として入力しているので、背景色が白になっている点に注目してください）。

Q : このカフェテリアには23個のリンゴがありました。そのうち 20 個をランチで使てしまい、6 個を購入しました。リンゴは何個ありますか？

A : ステップバイステップで考えてみましょう。

1. リンゴの最初の数を 23 個とします。
2. ランチで使った 20 個を引いて、3 個残ります。
3. 6 個を購入しましたので、計 9 個になります。

回答：現在、リンゴは9個あります。

答えを求めるプロンプト

このプロンプトを見ると、few-shot-CoTプロンプトと同様に見えます（ただし、入力と出力の例があるわけではなく、解きたい問題とその解法がプロンプトに含まれるだけです）。

概念的にはこのような2つのプロンプトをモデルに入力するのですが、実際には「ステップバイステップで考えてみましょう」と書くだけで、これらをまとめて行ってくれるというわけです。

手を煩わせる必要がないという点では、zero-shot-CoT プロンプトはかなり優秀ですが、言語モデルが正しく推測してくれないこともあります。そうしたときには few-shot-CoT で言語モデルの振る舞いを細かく制御するというのがよいでしょう（これについては次節で見てみます）。

ChatGPT はどうか

本稿の冒頭でも見たように、ChatGPT は少し複雑な（ステップが複数ある）計算問題についてはうまい具合に答えてくれます。

し

このカフェテリアには23個のリンゴがあります。そのうちの20個を使ってランチを作りました。その後、6個を買い足しました。リンゴは何個あるでしょう？



最初は23個のリンゴがあり、そのうち20個を使ったので、残りのリンゴは $23 - 20 = 3$ 個です。

その後6個を買い足したので、合計で $3 + 6 = 9$ 個のリンゴがあります。

少し複雑な計算問題なら ChatGPT は解いてくれる（ことが多い）

今度は、コインの表向き／裏向きをモデルに推測させてみます。

初期条件はコインが表向きに置いてあるものとして、以下のようにコインをひっくり返すとしましょう。かつて内はコインの向きがどう変化するのかをメモしたものです。

- ・ A 氏は 1 回ひっくり返す（表→裏）
- ・ B 氏は 2 回ひっくり返す（裏→表→裏）
- ・ C 氏はひっくり返さない（裏向きのまま）
- ・ その後、3 人が 1 回ずつひっくり返す（裏→表→裏→表）

というわけで、最終的にコインが表向きとなっているかどうかを ChatGPT が推測できるかを試してみます（正しい答えが得られることもあるでしょう）。

し

コインが表向きに置かれています。A氏はコインを1回ひっくり返しました。B氏はコインを2回ひっくり返しました。C氏はコインをひっくり返しませんでした。その後、3人がコインを1回ずつひっくり返しました。現在、コインは表向きになっているでしょうか？



現在、コインは「裏」向きになっています。

最初にコインが表向きに置かれていたとして、A氏が1回ひっくり返したため、コインは「裏」向きになります。その後、B氏が2回ひっくり返したため、コインはまた「表」向きになります。そして、C氏がコインをひっくり返さなかったため、コインは「表」向きのままになります。

最後に、3人全員がコインを1回ずつひっくり返した場合、コインは「裏」向きになります。したがって、現在のコインは「裏」向きになっています。

コインの表裏の推測は得意でないもよう

この結果を見ると、B氏がコインを2回ひっくり返すと、コインは裏向きから表向き、そしてまた裏向きになるはずですが、そこを理解できませんでした。そこを間違えたばかりにコインは最終的に裏向きだと言語モデルは推測をしました。

こんなときには zero-shot-CoT プロンプトの出番です。といっても、InstructGPT とは UI が異なっているので、ここでは入力プロンプトの最後に「ステップバイステップで考えてみましょう」と付加することにします。

その結果が以下です。

し

コインが表向きに置かれています。A氏はコインを1回ひっくり返しました。B氏はコインを2回ひっくり返しました。C氏はコインをひっくり返しませんでした。その後、3人がコインを1回ずつひっくり返しました。現在、コインは表向きになっているでしょうか？
ステップバイステップで考えてみましょう。



はい、ステップバイステップで考えてみます。

1. 最初にコインが表向きに置かれているため、表が上になっています。
2. A氏がコインを1回ひっくり返すため、裏が上になります。
3. B氏がコインを2回ひっくり返すため、表が上になります。
4. C氏がコインをひっくり返さないため、表が上になったままで。
5. 最後に、3人全員がコインを1回ずつひっくり返すため、コインが裏返ります。つまり、表が下になり、裏が上になります。

したがって、現在のコインは「裏」向きになっています。

ステップバイステップで間違えた

ここでは先ほどと同じポイント（B氏が2回ひっくり返した）で先ほどと同じ間違いをしたので、結果も間違ったものになってしまいました。

こうなったら few-shot-CoT プロンプトを試してみることにしましょう。ただし、ここでは現在の問題を解く手順をそのまま入力プロンプトに組み込むだけとします（他の入力と出力の組はここでは省略します）。

し

コインが表向きに置かれています。A氏はコインを1回ひっくり返しました。B氏はコインを2回ひっくり返しました。C氏はコインをひっくり返しませんでした。その後、3人がコインを1回ずつひっくり返しました。現在、コインは表向きになっているでしょうか？

- ・ A氏は1回ひっくり返すと表向きのコインが裏向きになります（表一裏）
- ・ B氏は2回ひっくり返すので、裏向きのコインが表向きになった後、それが裏向きになります（裏一表一裏）
- ・ C氏はひっくり返さないので裏向きのままで（裏）
- ・ その後、3人が1回ずつひっくり返すので裏向きのコインが表向きになり（1人目）、表向きのコインが裏向きになり（2人目）、最後に裏向きのコインが表向きになります（3人目）（裏→表一裏一表）



したがって、現在コインは表向きになっています。

few-shot-CoT プロンプト

自分でほぼ解答を入力しているので、ここでは正しい答えが出力されました。



言語モデルに推測させるというよりも、言語モデルと一緒に答えを考えている感が非常に強くあります（笑）。

このように ChatGPT は InstructGPT よりも正しい答えを出してくれることが多いのですが、うまくいかないときには zero-shot-CoT プロンプトを使って「ステップバイステップで考えてみましょう」としてみて、それでもダメなら few-shot-CoT プロンプトを試してみるというのは確かによいアプローチのようです。



「一緒に答えを考えている感」はなるほど。思考の連鎖によるプロンプトは、どうしても人間が子供に勉強というか質問への考え方を教えているのに似ているという感じがありますね。そういう意味では ChatGPT は、何でも答えてくれる大先生というよりも、すごく物知りでクリエイティブな発想ができる子供だと思って接するのがよいのかなと思いました。

あと、前回も同じ感想だったのですが、ますます人間が考え込むような論理的思考が必要となる質問への回答は、現状の ChatGPT にはあまり向いていないなと思いました。答えが間違っている可能性を考慮する必要があるため、結局は答え合わせを質問者がしなければならなくなるので。

というわけで、ChatGPT のプロンプトについてはこのくらいにして、次回は ChatGPT API を使って何かをしてみようと思います。

ChatGPT の API を使ってみよう： コンソールで対話するコードとは？

2023 年 3 月にリリースされた ChatGPT の API の使い方、API の振る舞い、対話をどのように管理するのか、コンソールで API を介して対話をするコードなどを紹介します。

かわさきしんじ, Deep Insider 編集部 (2023 年 03 月 24 日)

ChatGPT API

2023 年 3 月 1 日、OpenAI が ChatGPT の API をリリースしました。今回はこの API を実際に使って、その特徴を調べてみることにします。



さらに 2023 年 3 月 14 日には GPT-4 がリリースされましたね。GPT-4 は ChatGPT でも使用できますが、筆者はまだ待機リストに登録された状態なので、今回は gpt-3.5-turbo を言語モデルとして使用します（かわさき）。

[ChatGPT API のドキュメント](#)によれば、この API は一連のメッセージを入力として受け取り、モデルが生成したメッセージを出力とするとのことです。

API 呼び出しには次のようなものが必要になります。この連載の第 2 回でも説明しましたが、それとほぼ同じです（使用するクラスが openai.Completion から openai.ChatCompletion に変わっているところと、create メソッドで単なるプロンプトではなく、メッセージを要素とするリストを渡すところが異なる点です）。もう一度ここで簡単にまとめておきましょう。

- OpenAI へサインアップする
- OpenAI のサイトで API キーを取得する
- openai モジュールのインストールする (`pip install openai`)
- `openai.ChatCompletion.create` メソッドを呼び出す

API キーを取得する詳しい手順などは第 2 回を参照してください。また、openai モジュールを使うことで Python からこの API を呼び出せます。

コード的には次のようにとてもシンプルです。

```
import openai

KEY = "取得したAPIキー"
openai.api_key = KEY

messages = [
    # ChatGPTとの対話内容:
    # {"role": ロール, "content": メッセージ} という辞書を要素とするリスト
]

completion = openai.ChatCompletion.create(
    model="gpt-3.5-turbo", # ChatGPT API を使用するには 'gpt-3.5-turbo' などを指定
    messages=messages
)

print(completion)
```

ChatGPT API を呼び出すコードのひな型

以下では messages リストの内容についてお話をした後に、実際に API を呼び出してみます。

ChatGPT API を呼び出してみよう

第 2 回で取り上げた InstructGPT では openai.Completion.create メソッドを呼び出すときに prompt 引数にプロンプトを指定していました。第 2 回では次のようなコードを紹介していました。

```
response = openai.Completion.create(
    model="text-davinci-003", # InstructGPT
    prompt="晴れた日曜日の午後には何をすればいいかな?",
    # .....省略.....
)
```

InstructGPT API を呼び出すコードの例

これに対して、OpenAI のドキュメントでは ChatGPT API を呼び出すコード例としては以下が紹介されています。

```
openai.ChatCompletion.create(  
    model="gpt-3.5-turbo",  
    messages=[  
        {"role": "system", "content": "You are a helpful assistant."},  
        {"role": "user", "content": "Who won the world series in 2020?"},  
        {"role": "assistant", "content": "The Los Angeles Dodgers won the  
World Series in 2020."},  
        {"role": "user", "content": "Where was it played?"}  
    ]  
)
```

ChatGPT API を呼び出すコードの例

2つのコード例の差は、create メソッドの prompt 引数と messages 引数に現れています（model 引数の値も違っていますが）。InstructGPT API の prompt 引数は InstructGPT に対して何をしてほしいのかを指示するプロンプトです。対して、ChatGPT API の messages 引数はいわば人間と ChatGPT との間で行われる対話で送受信されるメッセージをリストに含めたものといえます。上のリストを見れば、リストの要素となっている辞書の "content" キーの値がそれらのメッセージを表していることは分かります。

では、"role" キーの値は何を表しているのでしょうか。その値としては "system" か "role" か "assistant" のいずれかを指定します。"system" は「ChatGPT API と会話を始めるときに、ChatGPT がどんなふうに振る舞うかを指定するときに使う」と考えておきましょう。その後、「人間が ChatGPT にメッセージを投げかけるときには "role" に "user" を指定」します。"assistant" は対話を続けるのに必要な情報を含めるのですが、これについては後で見てみましょう。

何はともあれ、上のメッセージリストを使って API を呼び出してみましょう。以下に示すコードは、API からの戻り値を変数 `completion` に取っておくようにしたり、リストを変数 `messages` に代入したりしていますが、やっていることは上で紹介した API 呼び出しと同じです。

```
import openai

KEY = "取得したAPIキーをここに記述"
openai.api_key = KEY

messages = [
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": "Who won the world series in 2020?"},
    {"role": "assistant", "content": "The Los Angeles Dodgers won the World Series in 2020."},
    {"role": "user", "content": "Where was it played?"}
]

completion = openai.ChatCompletion.create(
    model="gpt-3.5-turbo",
    messages=messages
)

print(completion)
```

ChatGPT API を呼び出してみる

`openai.ChatCompletion.create` メソッドを呼び出す際に、`model` 引数に "gpt-3.5-turbo" と指定していますが、ChatGPT API を呼び出す際にはこれを指定します。なお、ChatGPT で GPT-4 を使用するのであれば "gpt-4" を指定します。モデルの詳細については OpenAI のドキュメント「[Models](#)」を参照してください。特定の日付が付いたスナップショットもリリースされています ("gpt-3.5-turbo-0301" や "gpt-4-0314" など)、"gpt-3.5-turbo" と "gpt-4" を指定した場合は API を呼び出した時点で最新の言語モデルが使われます。日付付きはその特定の日付で指定された言語モデルが使われます。

以下はこれを Visual Studio Code (以下、VS Code) で実行した結果です。

The screenshot shows a Python script running in VS Code. The code sends a message to ChatGPT asking about the 2020 World Series and receives a response. The execution time is 2.9 seconds.

```
messages = [
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": "Who won the world series in 2020?"}, 
    {"role": "assistant", "content": "The Los Angeles Dodgers won the World Series in 2020."}, 
    {"role": "user", "content": "Where was it played?"}
]

completion = openai.ChatCompletion.create(
    model="gpt-3.5-turbo",
    messages=messages
)

print(completion)
[2] ✓ 2.9s
```

... {
 "choices": [
 {
 "finish_reason": "stop",
 "index": 0,
 "message": {
 "content": "The 2020 World Series was played in Globe Life Field in Arlington, Texas due to the COVID-19 pandemic",
 "role": "assistant"
 }
 }
],
 "created": 1679398865,
 "id": "chatcmpl-6wUePzWigANngd1aeOOrRlaAeWvQn",
 "model": "gpt-3.5-turbo-0301",
 "object": "chat.completion",
 "usage": {
 "completion_tokens": 25,
 "prompt_tokens": 56,
 "total_tokens": 81
 }
}

呼び出し結果

戻り値の先頭要素 ("choices" 要素) の値はリストになっていますが、筆者が試した限りではその要素は辞書 (と同様に使えるオブジェクト) が 1 つだけとなっています (将来変更される可能性はあります)。その "message" キーの値 (completion["choices"][0]["message"]) に ChatGPT が生成したテキストが含まれています ("content" キーの値)。

生成されたテキストは、上の例では「The 2020 World Series was played in Globe Life Field in Arlington, Texas due to the COVID-19 pandemic」となっています。これは messages リストの最後の要素にある「Where was it played?」(それはどこで開催されましたか?) に対応する返答です。

ここで「ん?」と思いませんか? messages 引数には幾つかのメッセージが含まれているのに、返送されたテキストは 1 つだけです。「Who won the world series in 2020?」(2020 年のワールドシリーズで優勝したのはどこ?) に対応する返答はありません。

これから予想されるのは、ChatGPT API は対話の最後に生成したメッセージだけを返信するということです。対話の途中で生成されたテキストは全て捨てられているようです。API がそういう仕様であれば、メッセージを 1 つずつ送信すればよさそうな気もします。次にこれを試してみましょう。

ChatGPT API は対話内容を記憶しない

messages リストには合わせて 4 つのメッセージが含まれていました（そのうちの 1 つは "role" が "assistant" になっていますが、取りあえず、そこは気にしないことにします）。これを順次 API に渡して、その戻り値を表示するコードを以下に示します。

```
messages = [
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": "Who won the world series in 2020?"},
    {"role": "assistant", "content": "The Los Angeles Dodgers won the World Series in 2020."},
    {"role": "user", "content": "Where was it played?"}
]

for msg in messages:
    completion = openai.ChatCompletion.create(
        model="gpt-3.5-turbo",
        messages = [msg]
    )
    print(completion["choices"][0]["message"]["content"])
```

対話の内容を 1 つずつ ChatGPT API に投げるようにしたコード

このコードでは for 文を使って messages リストに含まれている要素（辞書）を 1 つずつ取り出して、それを messages 引数に（リストに詰め込んで）渡すようにしています。それから ChatGPT が生成したテキストだけを画面に表示するようにしました。残りは先ほどのコードと同じです。

実行結果は次のようになります。

```
messages = [
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": "Who won the world series in 2020?"},
    {"role": "assistant", "content": "The Los Angeles Dodgers won the World Series in 2020."},
    {"role": "user", "content": "Where was it played?"}
]

for msg in messages:
    completion = openai.ChatCompletion.create(
        model="gpt-3.5-turbo",
        messages = [msg]
    )
    print(completion["choices"][0]["message"]["content"]) # type: ignore

[3]   ✓  29.8s
```

...

Thank you! How may I assist you?

The Los Angeles Dodgers won the World Series in 2020.
That is correct! The Dodgers defeated the Tampa Bay Rays in six games to win their first World Series championship since 1981.

I'm sorry, I cannot determine what you are referring to. Could you please provide more context or information?

最後の返事に注目

messages リストには 4 つのメッセージがあったので、返答も 4 つあります。最初の返答は「You are a helpful assistant」（あなたはとても有能なアシスタントです）に対するもので「Thank you! How may I assist you?」（ありがとうございます。何かお手伝いしましょうか？）となっています。いい感じです。

次の返答は「Who won the world series in 2020?」（2020 年のワールドシリーズで優勝したのはどこ？）に対するものです。「The Los Angeles Dodgers won the World Series in 2020」（ドジャースが優勝）という答えになっていますね。

その次は「The Los Angeles Dodgers won the World Series in 2020」（2020 年のワールドシリーズではドジャースが優勝しました）となっているので返答はそれを肯定するものになっています（"role" が "assistant" になっていることについては後で説明します）。

問題は最後の返答です。「Where was it played?」つまり「それはどこで開催されましたか？」というメッセージに対して、「I'm sorry, I cannot determine what you are referring to. Could you please provide more context or information?」（すみません。何のことをいっているのか判断が付きません。もっと情報を与えてくれますか？）と返されています。4 つのメッセージをまとめてリクエストしたときには開催地を答えてくれたのにそうはいきません。

これは ChatGPT が複数回の API 呼び出しの間で、対話に関する情報（記憶、コンテキスト）を維持しないためです。この場合は「2020 年のワールドシリーズでドジャースが優勝した」という情報が最後の API 呼び出しではなくなっているからです。文脈がないところで「それ」「it」といわれても ChatGPT は困ってしまうというわけです。

ChatGPT の Web UI を例に考えてみましょう。メッセージをひとまとめにして API に送るというのは、以下のような連続したやりとりを行って、最後の返答だけを返してきたものだと考えられます。

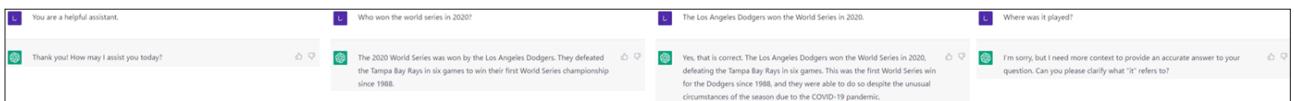
The screenshot shows a series of messages in a chat interface. The messages are as follows:

- User message: You are a helpful assistant.
- ChatGPT response: Thank you! How may I assist you today? (with like/dislike buttons)
- User message: Who won the world series in 2020?
- ChatGPT response: The Los Angeles Dodgers won the World Series in 2020. (with like/dislike buttons)
- User message: The Los Angeles Dodgers won the World Series in 2020.
- ChatGPT response: Yes, that's correct! The Los Angeles Dodgers won the World Series in 2020. It was their first World Series win since 1988. (with like/dislike buttons)
- User message: Where was it played?
- ChatGPT response: The 2020 World Series was played at Globe Life Field in Arlington, Texas. This was the first time in modern history that the entire World Series was played at a neutral site. (with like/dislike buttons)

4 往復の対話（イメージ）

このときには、対話を続けていく中で出てきた話題や情報を ChatGPT が覚えており、「it」が何を指しているのか、といったことも文脈に応じて判別できています。

一方、今のようにメッセージをバラバラにして ChatGPT API を呼び出すのは、上のような連続した対話にはならず、一往復の対話を 4 回続けただけということです。以下の画像はその例ですが、画面サイズを縮小しているので文字が読みづらくなっています。が、ここでは文字の内容はどうでもよいです。一往復のリクエストが 4 回というイメージをつかんでください。



一往復の対話を 4 回行う（イメージ）

というわけで、一往復の対話（API 呼び出し）をしながら、ユーザーの入力ごとに API からの返答を表示するにはどうしたらよいでしょう。



ChatGPT の Web UI では [New chat] で新規のチャットが作られてその対話の流れ全てが履歴に残りますが、対話の流れ全てを覚えている必要があるからなのだと合点がいきました。

コードは少し長くなって面倒な印象でしたが、対話の流れ全てをメッセージとして渡すためには仕方がないですね（一色）。

messages リストで対話内容を管理する

「ユーザーが入力して返答を受け取り表示する」という一往復の対話を繰り返しながら、さも ChatGPT API と流れるように対話を行うには以下のようなことを行います。かっこ内の「"system" メッセージ」などは "role" キーに指定する値を表しています。"system" / "user" / "assistant" のいずれかになることを先ほどお話しした通りです。

1. messages リストに最初のメッセージを格納する ("system" メッセージ)
2. ユーザーからの入力をリストに追加する ("user" メッセージ)
3. API を呼び出す
4. API から返答をリストに追加する ("assistant" メッセージ)
5. ユーザーからの入力をリストに追加する ("user" メッセージ)
6. API を呼び出す
7. API から返答をリストに追加する ("assistant" メッセージ)
8. 対話が終わるまでこれを繰り返す

このようにしてユーザーが何を入力したか、API が何を返したかを messages リストに全て保管して、API を呼び出すときにそれらをまとめて渡すようにします。

以下のコードを見てください。

```
user_inputs = [
    {"role": "user", "content": "Who won the world series in 2020?"},
    {"role": "user", "content": "Where was it played?"}
]

messages = [
    {"role": "system", "content": "You are a helpful assistant."}
]

for user_input in user_inputs:
    print(f'>>> {user_input["content"]}')
    messages.append(user_input)
    completion = openai.ChatCompletion.create(
        model = 'gpt-3.5-turbo',
        messages=messages
    )
    content = completion["choices"][0]["message"]["content"] # type: ignore
    print(content)
    messages.append({"role": "assistant", "content": f'{content}'})
```

messages リストで対話内容を管理する

ここではユーザー入力を模したものを `user_inputs` リストに含めるようにして、`messages` リストには最初は `"system"` メッセージだけを含めるようにしました。`"assistant"` メッセージがないのは、API からの返答を `messages` リストに追加するためです。

その後は、`user_inputs` リストの内容を `messages` リストに追加するたびに、API を呼び出して、その返答を表示して、リストに追加しているだけです（最初の `"system"` メッセージの表示は省略しています）。

このコードの実行結果を以下に示します。

```
user_inputs = [
    {"role": "user", "content": "Who won the world series in 2020?"},  
    {"role": "user", "content": "Where was it played?"}
]  
  
messages = [  
    {"role": "system", "content": "You are a helpful assistant."}  
]  
  
for user_input in user_inputs:  
    print(f'>>> {user_input["content"]}')  
    messages.append(user_input)  
    completion = openai.ChatCompletion.create(  
        model = 'gpt-3.5-turbo',  
        messages=messages  
    )  
    content = completion["choices"][0]["message"]["content"] # type: ignore  
    print(content)  
    messages.append({"role": "assistant", "content": f'{content}'})  
  
[4] ✓ 11.0s  
... >>> Who won the world series in 2020?  
The Los Angeles Dodgers won the World Series in 2020.  
>>> Where was it played?  
The World Series in 2020 was played in Arlington, Texas.
```

対話の内容が毎回表示されるようになった

"user" メッセージは「>>>」というプロンプト付きで表示するようにしたので、いかにも人間と ChatGPT が対話しているように見えますね。

この状態で `messages` リストがどうなっているか見てみましょう。

```
[5] ✓ 0.1s  
for msg in messages:  
    print(msg)  
  
... {'role': 'system', 'content': 'You are a helpful assistant.'}  
{'role': 'user', 'content': 'Who won the world series in 2020?'}  
{'role': 'assistant', 'content': 'The Los Angeles Dodgers won the World Series in 2020.'}  
{'role': 'user', 'content': 'Where was it played?'}  
{'role': 'assistant', 'content': 'The World Series in 2020 was played in Arlington, Texas.'}
```

`messages` リストの内容

最後の API 呼び出しの結果も含まれているので 5 行になっていますが、最初の例で見たものと同様な要素が含まれていることが分かります（最初の例の `messages` リストも似たような方法で作成したものだと筆者は予想しています）。



上の画像を見ると、2往復の対話が一度のAPI呼び出しで行われているように見えますが、そうではなくAPI呼び出しが2回行われていることには注意してください。この例では過去のイベントの結果を問い合わせているので、一貫性のある会話となっていますが、1回目の呼び出しで行われた対話の内容が2回目の内容と大きく変化して、対話の内容がかみ合っていないように見える可能性もあります。

コンソールでチャットする

messagesリストで対話内容を管理して、APIにそれを渡せばよいことが分かったので、コンソールでChatGPTと対話をするためのコードを紹介します。

```
import openai
KEY = "取得したAPIキーをここに記述"
openai.api_key = KEY

class ChatManager:
    def __init__(self, model, sysmsg="こんにちは、アシスタントさん!"):
        self.model = model
        self.messages = []
        self.sysmsg = sysmsg
        self.compose_message("system", self.sysmsg)
        self.params = {}
        self.setup_params()

    def compose_message(self, role, content):
        self.messages.append({"role": role, "content": content})

    def setup_params(self, **kwargs):
        self.params["model"] = self.model
        self.params["messages"] = self.messages
        keys = kwargs.keys()
        key_list = ["temperature", "top_p", "n", "stream", "stop", "max_tokens",
                   "presence_penalty", "frequency_penalty", "logit_bias", "user"]
        for key in key_list:
            if key in keys:
```

```

        self.params[key] = kwargs[key]

def call_chat_api(self):
    completion = openai.ChatCompletion.create(**self.params)
    result = completion["choices"][0]["message"] # type: ignore
    return result

def get_response_from_api(self):
    result = self.call_chat_api()
    print(result["content"])
    self.compose_message(result.role, result.content)

def chat_by_api(self, content):
    self.compose_message("user", content)
    print(f">>> {content}")
    self.get_response_from_api()

def chat_on_console(self):
    print("チャットを始めます。終わるときには「exit」と入力してください。")
    content = input(">>> ")
    while content != "exit":
        if content == "show status":
            print(self.messages)
            content = input(">>> ")
            continue
        self.compose_message("user", content)
        self.get_response_from_api()
        content = input(">>> ")

if __name__ == "__main__":
    chatmgr = ChatManager("gpt-3.5-turbo")
    chatmgr.chat_on_console()
    print("exit chat")

```

ChatMnager クラス

詳細なことには触れませんが、ここでは ChatManager クラスに messages リストとそれを管理するコード、ChatGPT API 呼び出しを行うコードなどをまとめてあります。setup_params メソッドでは create メソッド呼び出して指定可能なパラメーターを設定できるようにしました（が、本稿では使いませんでした）。

chat_on_console メソッドがコンソールで対話するメインコードです。ここではコンソールから入力された値を受け取って、それを基に messages リストに "user" メッセージを追加し、API を呼び出して、その結果を使って messages リストを更新する処理を行っています。このときには、messages リストの内容を表示したり、「exit」と入力すると対話を終了したりするようにしてあります。



VS Code 上の Jupyter ノートブックで対話をできるようなメソッドも含めてあります
(Jupyter で試してからコンソールに話を移す予定だったのですが、文字数的にあきらめました)。

実行している様子を以下に示します。

```
PS C:\chatgpt\chatgpt_06\test> & c:/chatgpt/chatgpt_06/test/venv/Scripts/Activate.ps1
○ (venv) PS C:\chatgpt\chatgpt_06\test> python .\chatmgr.py
チャットを始めます。終わるときには「exit」と入力してください。
>>> 祝日のよい過ごし方を教えてください
はい、いくつかのアイデアをご紹介します！

1. 外出する：天気が良ければ、公園や自然の中に出かけて、散歩やピクニックを楽しんでみましょう。
2. 読書や映画鑑賞：家で過ごす場合は、読書や映画鑑賞といった、ゆっくりとした時間が過ごせる活動がおすすめです。自分の興味のあるジャンルを選び、好きなものを楽しんでみましょう。
3. ゲームをする：ゲーム好きな方は、自分の好きなゲームで時間を過ごすのもオススメです。熱中できるゲームであれば、気分転換にもなります。
4. 趣味に没頭する：趣味や特技がある場合は、祝日を有意義に過ごすために、その趣味に没頭するというのもいいでしょう。編み物や料理、楽器演奏など、自分が得意とすることで自己成長も促進できます。

以上、いくつかのアイデアをご紹介しました。いずれも、自分なりの過ごし方を見つけて、リフレッシュすることが大切です。
>>> 僕はずっと原稿を書いていたのですが、これはいい過ごし方なんですかね
原稿を書くことは、とても充実した過ごし方の一つです。自分の思考を整理し、表現力を磨くことができます。また、書くことによってストレス発散にもなります。しかも、書き上げたときには達成感が得られるので、やりがいも感じられます。ただし、長時間ずっと書き続けると、身体に負担がかかる場合があるので、こまめに休憩をとることが大切です。適度な休息をとりながら、時間をかけて書き上げることが大切です。
>>> show status
[{'role': 'system', 'content': 'こんにちは、アシスタントさん！'}, {'role': 'user', 'content': '祝日のよい過ごし方を教えてください'}, {'role': 'assistant', 'content': 'はい、いくつかのアイデアをご紹介します！\n\n1. 外出する：天気が良ければ、公園や自然の中に出かけて、散歩やピクニックを楽しんでみましょう。\n\n2. 読書や映画鑑賞：家で過ごす場合は、読書や映画鑑賞といった、ゆっくりとした時間が過ごせる活動がおすすめです。自分の興味のあるジャンルを選び、好きなものを楽しんでみましょう。\n\n3. ゲームをする：ゲーム好きな方は、自分の好きなゲームで時間を過ごすのもオススメです。熱中できるゲームであれば、気分転換にもなります。\n\n4. 趣味に没頭する：趣味や特技がある場合は、祝日を有意義に過ごすために、その趣味に没頭するというのもいいでしょう。編み物や料理、楽器演奏など、自分が得意とすることで自己成長も促進できます。\n\n以上、いくつかのアイデアをご紹介しました。いずれも、自分なりの過ごし方を見つけて、リフレッシュすることが大切です。'}, {'role': 'user', 'content': '僕はずっと原稿を書いていたのですが、これはいい過ごし方なんですかね'}, {'role': 'assistant', 'content': '原稿を書くことは、とても充実した過ごし方の一つです。自分の思考を整理し、表現力を磨くことができます。また、書くことによってストレス発散にもなります。しかも、書き上げたときには達成感が得られるので、やりがいも感じられます。ただし、長時間ずっと書き続けると、身体に負担がかかる場合があるので、こまめに休憩をとることが大切です。適度な休息をとりながら、時間をかけて書き上げることが大切です。'}]
>>> exit
```

VS Code のコンソールで対話をしているところ

ChatGPT API を介して ChatGPT の言語モデル（gpt-3.5-turbo）と対話をしているように見えますね。

ChatGPT の動作がある程度分かれば、これを活用した Web アプリなどを構築するのもそれほど難しいことではないでしょう。



どうすれば作れるのかを ChatGPT に聞いてみればよいのでは!! (笑)



ChatGPT に聞いてみましたが、最新の API は知らないみたいでした。残念。

とはいっても、基礎的なお話はできたので、ここから先は読者の皆さんにお任せしようと思います。



編集:@IT 編集部
発行:アイティメディア株式会社
Copyright © ITmedia, Inc. All Rights Reserved.