生成 AI を使い倒すためのプロンプトエンジニアリングとは

～大規模言語モデル（LLM）の性能を正しく引き出せるかどうかは利用者の指示力にかかっている～

株式会社第一生命経済研究所　客員研究員　片柳　宏太
（第一生命情報システム株式会社　デジタル推進部所属）

（要旨）

○　近年、大規模言語モデル（LLM）の進化が激しさを増している。特に生成 AI である ChatGPT の登場は世間に大きな衝撃を与え、いたるところで活用検討が行われるようになっている。

○　LLM をはじめとする AI の活用には、モデルを個別の目的に特化させなければならないところに課題がある。モデルのカスタマイズ手法として、モデル自体の性能が大幅に向上してきた現在において、比較的容易に取り組める「プロンプトエンジニアリング」に注目が集まっている。

○　プロンプトエンジニアリングとは、LLM の利用者がモデルに投入する指示や質問を最適化し、モデルを効率的に使用する技術のことである。モデルに対して指示や質問をするための文章や言葉のことを「プロンプト」と呼び、プロンプトエンジニアリングではこの「プロンプト」を達成したい目的に適した形式や内容に調整していく作業が行われる。

○　プロンプトを最適化することによって、LLM は幅広い分野や用途での活用が期待できるようになる。また、プロンプトの良し悪しがモデルの出力に大きな影響を与える。適切なプロンプトを構築することにより、LLM の精度向上や領域拡張の効果が期待できるようになる。

○　今後、AI のさらなる進化が想定されるが、そのポテンシャルを十二分に発揮できるかは利用者の指示に掛かっている。その根幹となる「プロンプトエンジニアリング」について理解を深め、個別のユースケースに応用していくことが今後の活用における重要な要素となるだろう。

1. 急速に進化する大規模言語モデル（LLM）

近年、ChatGPT をはじめとした自然言語処理分野における汎用的な大規模言語モデル（LLM：Large Language Model）（以下、LLM と呼ぶ）の進化が勢いを増している。LLM は、膨大な量のテキストデータを学習した統計モデルで、言語の構造や文脈を理解し、複雑な処理（文書生成、質問応答、文章要約、言語翻訳、文章分類、感情分析

など）を実行することができる。

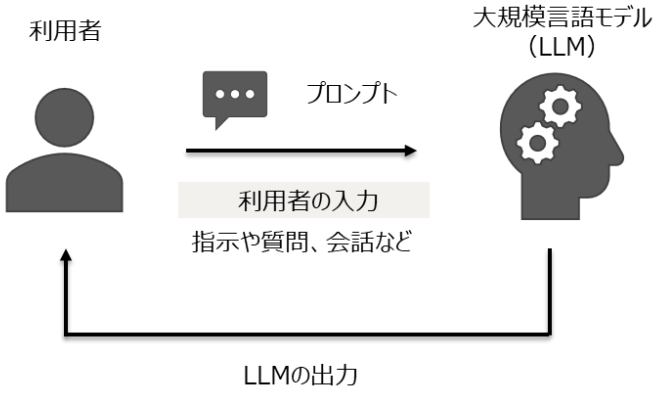
2022 年 11 月 30 日に OpenAI 社の生成 AI である ChatGPT が公開され、その性能と活用領域の広さから過去に類を見ないほど急激な拡大をみせている。ChatGPT は公開からわずか 2 ヶ月間で利用者数が 1 億人を超え、個人や企業のみならず国家レベルで活用策の検討やリスク統制等についての議論が行われている。

今後ますます進化していくであろう LLM を、利用する側においてはいかにして上手に使いこなすかが重要な課題となる。

2. 大規模言語モデルの利用とプロンプト

LLM の使い方はとてもシンプルである。LLM に対して処理させたい作業や答えて欲しい質問などを入力するだけである。例えば、LLM に「“This is a pen.”を日本語に翻訳して」と指示すると「これはペンです」と返してくれ、「日本で一番高い山は？」と質問すると「富士山」と返してくれる。このように、LLM は入力の内容に応じて様々な出力をすることが可能である（資料 1）。

資料 1 LLM を直接利用するイメージ



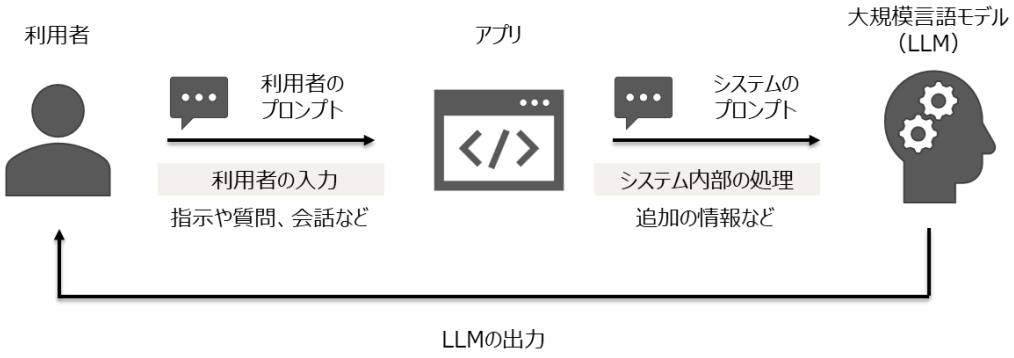
（出所）筆者作成

一般的にコンピューターや AI に対して、指示や質問をするための文章や言葉のことをプロンプトと呼ぶ。そのため、先程の例に登場した「“This is a pen.”を日本語に翻訳して」という指示や「日本で一番高い山は？」といった質問は全てプロンプトとなる。利用者はモデルへの入力となるプロンプトを作成し、LLM に投入することで簡単に利用することが可能となる。

ここで、ChatGPT のようにアプリを通して LLM を利用する場合、必ずしも利用者の作成したプロンプトがそのまま LLM に投入されるとは限らないことに注意が必要だ。これは、利用者が作成したプロンプトをもとに、システム内部で LLM に投入するためのプロンプトを作成している場合があるからである。例えば ChatGPT の応答は、利用

者の直近の指示や質問に関わらず「過去の会話のやり取り」も考慮したものとなっている。これは、利用者が作成したプロンプトにシステム内部で今までの会話履歴を含める処理が行われ、そのプロンプトがLLMへと投入されているからである。このように、LLMを何かしらのアプリを通して利用する場合においては、利用者が作成するプロンプトとアプリ開発者がシステム内部の処理として組み込んだプログラムによって作成されるプロンプトの2つが存在する可能性があることを認識しておきたい（資料2)。

資料2 アプリを通してLLMを利用するイメージ

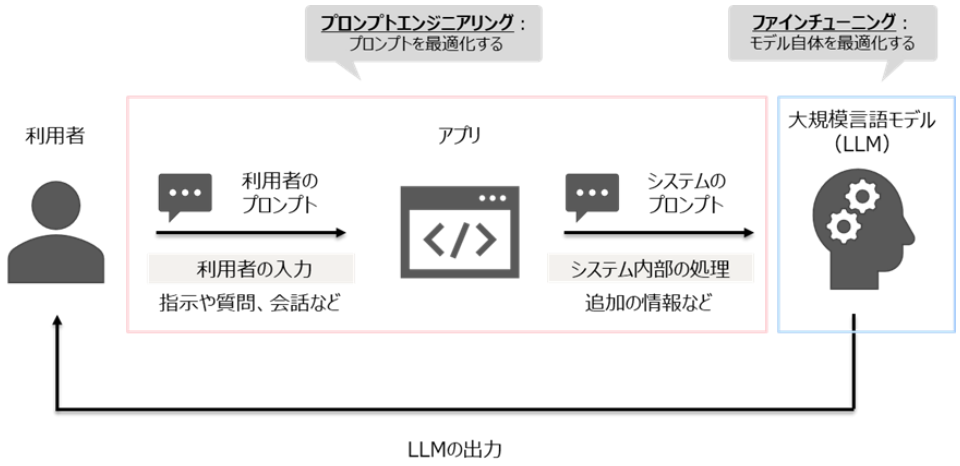


（出所）筆者作成

3. 大規模言語モデルの活用における課題

LLMの活用にあたっては、達成したい個別の目的にモデルをカスタマイズしなければならないといった課題がある。一般的に利用可能なLLMは汎用的に作られているものであり、利用者が求める個別の用途に合わせて作られたものではないからである。カスタマイズを行うことで、モデルの応答の質を高められるだけでなく、情報の追加やアップデートを行うことが可能となる。一般的に利用可能なLLMに対してカスタマイズを施す手法として、主に次の2つが存在する（資料3)。

資料 3　大規模言語モデルのカスタマイズ手法



（出所）筆者作成

1 つ目は達成したい目的に応じてモデルへの入力を最適化する「プロンプトエンジニアリング（Prompt Engineering、以下 PE）」である。PE とは、LLM の利用者がモデルへの入力である「プロンプト」を最適化し、モデルを効率的に使用する技術のことである。ここで言う LLM の利用者とは、主として自身が開発するアプリの中に LLM を組み込むアプリ開発者を想定しているが、PE 自体は（利用するアプリの仕様にも依るが）アプリの利用者を含めたすべての LLM 利用者が行えるものである。PE を行うことにより LLM が必要な情報や作業にフォーカスすることが可能となる。また、プロンプト内に LLM が学習していない情報や LLM の挙動に関する制御などを記載することで、LLM の性能や安全性の向上といった効果を得ることが可能となる。

2 つ目はモデル自体を特定の用途に対して最適化させる「ファインチューニン（Fine-tuning、以下 FT）」である。FT は、既存の LLM をベースに、用途に応じた追加のデータを用いてモデルを再学習する手法である。事前に学習していない知識や領域、特定の用途に対してモデルの性能を向上させることが可能である。各手法における特徴については、資料 4 を参照されたい。

資料 4 PE と FT の特徴

特徴	プロンプトエンジニアリング	ファインチューニング
目的	目的に沿った応答を生成する 確率を高める	特定のタスクや領域に対する 性能の向上
学習イメージ	都度渡される カンニングペーパーを学習	事前に専門の教科書や参考書を学習
LLM自体 の改変	無し	有り
メリット	✓ 応答の柔軟性が高い ✓ 応答の個別最適化に強い	✓ 事前に学習していない知識や領域においても追加の学習が可能 ✓ 難度の高いタスクにも適用可能
デメリット	✓ 入力で渡せるプロンプトの文字数に制限がある ✓ 応答に時間を要する ✓ 事前に全く学習していない領域には対応が難しい	✓ データやリソースの準備コストが大きい ✓ 最新情報や追加データの取り込みに時間が掛かる

（出所）筆者作成

以前は FT が主流であったが、ここ最近は PE の比重が高まっている。その背景には、ChatGPT で指摘されているように LLM 自体の性能が大幅に上がってきているということがある。簡易な用途であればモデル自体の最適化を行わなくても比較的高い精度で目的を達成できるようなった。また前提として、FT は大量データやモデルの再学習が必要となるため、実施にはある程度のスキルや作業負荷が必要となる。そのため、これらを必要とせず、利用者からの入力内容を調整するだけで実施が可能な PE でのカスタマイズに優先して取り組む傾向が強くなっている。

次節以降ではここ最近で注目を集めるようになった PE に着目し、より詳細な内容や具体的な活用事例について解説していく。

4. プロンプトエンジニアリングの必要性

改めて、PE とは「モデルの入力となるプロンプトの内容を最適化する技術」のことである。実際にプロンプトの良し悪しが、その後のモデルの出力に大きく関わってくる。このプロンプトとモデル出力の関係性については、上司が部下に対して指示を出す時とよく似ている。

その場合、指示の出し方 1 つで部下の成果が左右されることは容易に想像できるだろう。指示の内容が抽象的であったり、部下の能力や適性を考慮できていない場合は成果を著しく劣化させてしまう可能性がある。一方で、正しい手順で行えるように誘導したり、部下が持ち合わせていない知識やスキルを補うことで想定以上の成果を発

揮することも可能となる。

これと同様のことが LLM についても当てはまる。LLM にどのような指示（プロンプト）を与えるかによって、処理させたい作業を正しく実行できるかが変わってくる。もちろん、プロンプトを最適化することで全ての作業を正しく実行できるようになるわけではない。しかし、本来問題なく実行できる作業であるにもかかわらず、指示の出し方が十分でないことにより、正しい実行結果が得られないという問題をなくすことは可能である。また、LLM に不足している知識や機能をプロンプトで補うことにより、従来では答えることができない領域の課題についても処理が行えるようになる場合がある。

このように、プロンプトの良し悪しが LLM の出力に大きな影響を及ぼす。

5. プロンプトエンジニアリングの基礎

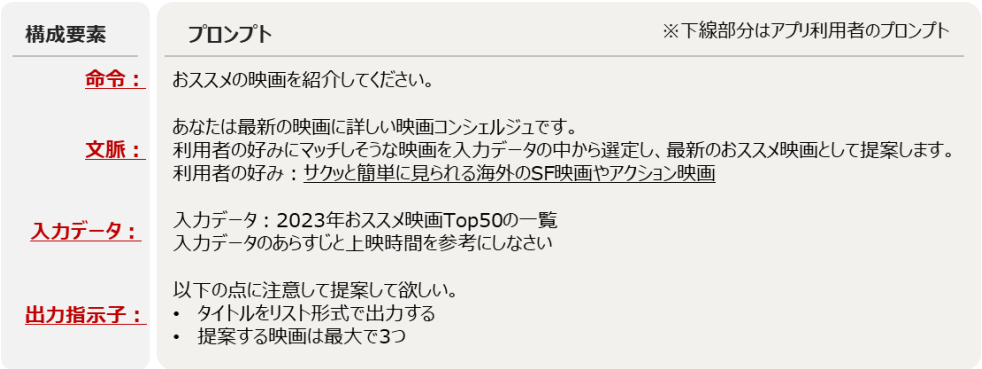
それでは、LLM にとっての良い指示とは一体どのようなものなのか。それを紐解いていく前に、まずはその大元となる「プロンプト」について理解を深めておく。

プロンプトを構成する要素は主に以下の 4 つがあり、用途によって様々なフォーマットが存在する。なお、必ず全ての要素が必要となるわけではない（資料 5）。

【プロンプトを構成する要素】

命令：モデルに実行して欲しいタスクまたは命令
文脈：モデルをより良い応答に導くための追加情報
入力データ：何かしらの処理を施したい入力や外部データ
出力指示子：出力のタイプや形式

資料 5 プロンプトを構成する要素のイメージ



（出所）筆者作成

「命令」はどの用途においても基本的に必須の要素である。「生成する」「分類する」

「要約する」「翻訳する」など、モデルにやらせたいことを指示する役割を持つ。また、「命令」以外の 3 つの要素は用途に応じて必要となる任意の要素である。「文脈」及び「入力データ」は、命令を実行する際の背景や目的、処理対象となるテキストなどを入力し、モデルの出力結果を利用者の意図に沿いやすくする役割を持つ。「出力指示子」は、出力に関する制約や出力形式（テキストやコードなど）を条件付ける役割を持つ。これら 4 つの要素を的確に指示することで、様々な用途に対する効果的なプロンプトを設計することが可能となる。

プロンプトの構成要素について理解を深めたところで、本題である LLM にとって良い指示にするために考慮すべきポイントを確認する。

【効率的なプロンプトを作成するためのポイント】

○命令

- 基礎的な質問文または指示を記述する。

○文脈

- 具体的な役割や目的を記述する。

○入力データ

- 処理を施すデータだけでなく、データをどのように扱って欲しいかを具体的に記述する。

○出力指示子

- 出力形式だけでなく、文字数や表現方法など出力に関する要望を出来るだけ詳細に記述する。
- 「しないこと」よりも「すること」を記述する。
- 複数の制約がある場合には 1 つずつ記述する。

以上の観点が、「LLM にとって望ましい指示」を作成する上で意識すべき重要なポイントと言われている。各要素の観点一つひとつに着目してみると、人間にとって良いとされる指示の特徴とほとんど同じであることが分かる。このことからも、LLM がしっかりと仕事をしてくれるかは、利用者の指示出しにかかっていると言えよう。

この他にも、Few-shot（注 1）、Chain-of-Thought（CoT）を初めとする様々な PE のテクニックが存在する（CoT についての解説は後述する）。これらのテクニックを駆使し、時には外部システムとの連携を上手に織り交ぜながら、いかにして利用者の要求を正しく処理してくれる指示を作るかが PE の醍醐味である。なお、今回は上記ポイントや PE のテクニックについて個別の解説は割愛するが、興味がある方は参考文献にて詳細をご確認いただきたい（DAIR.AI（2023））。

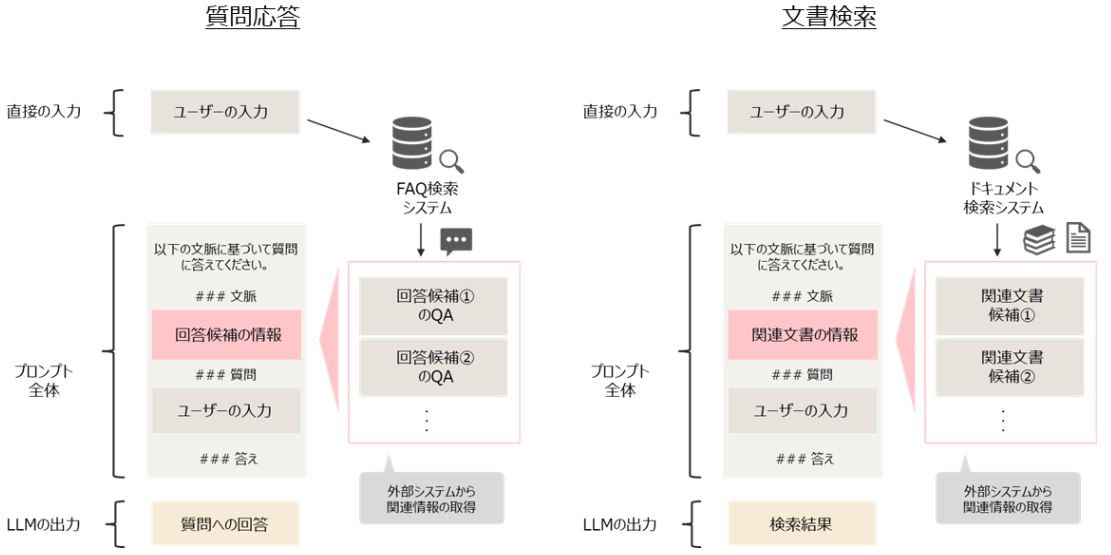
6. プロンプトエンジニアリングの活用事例

プロンプトを最適化することによって、LLM は幅広い分野や用途での活用が期待できるようになる。単なる会話や質問応答のみならず、分類、推論、抽出など、アイデア次第で本当に様々な用途に応用することが可能である。ここからは PE の活用事例についていくつかの具体例と共に確認する。

(1) 質問応答・文書検索

まずは LLM の活用において代表的なユースケースである「質問応答」と「文書検索」について必要なプロンプトを確認する。これらの機能を実現するには、当然のことながら利用者からの問い合わせに関する知識や文書の情報が必要となる。ここで 1 つ注意しておきたいのは、LLM 単体には事前に学習した知識以外に必要となる情報については持ち合わせていないという点である。個別の「質問応答」や「文書検索」においては、いずれも関連情報の抽出機能が必要となるが、LLM の機能としては備わっていないため、別途機能の作り込みが必要となる。そうして得られた情報をプロンプトに入力することで、「質問応答」や「文書検索」への適用が可能となる。「質問応答」や「文書検索」以外でも、必要に応じて外部システムから得られた情報をプロンプトに投入することで、LLM の活用領域を広げることが可能となる（資料 6)。

資料 6 質問応答及び文書検索を実現するために必要なプロンプト

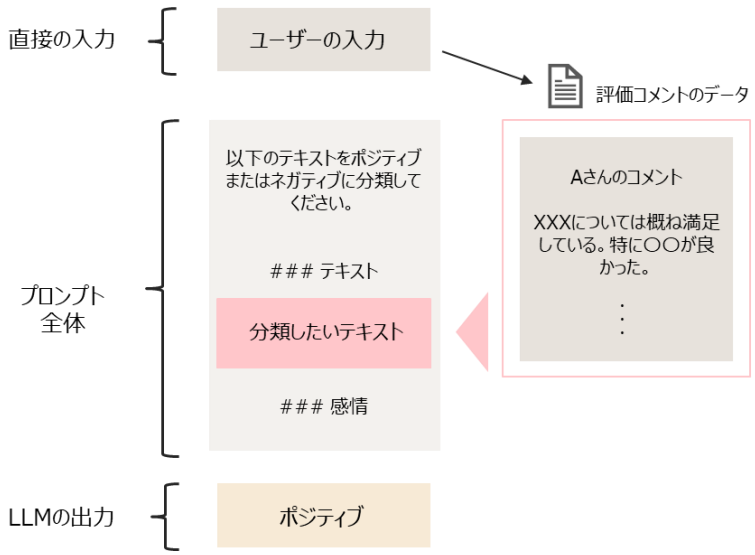


(出所)筆者作成

(2) 文章分類

次に文章分類での活用事例を確認する。最もシンプルな例としては資料 7 のようなプロンプトが考えられる。最低限「分類する対象」と「どのように分類するか」をプロンプトに記述することで文章分類への適用が可能となる。

資料 7　文章分類を実行するために必要なプロンプト

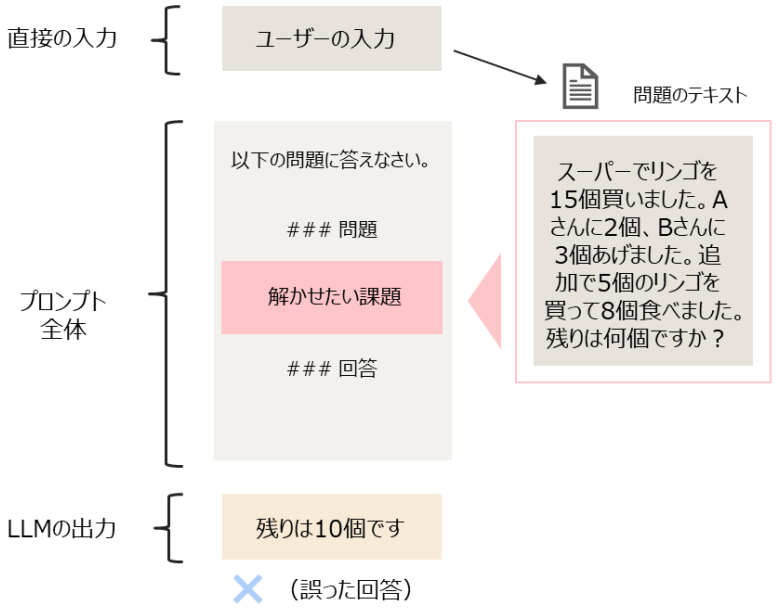


（出所）筆者作成

(3) 推論

3 つ目の活用事例として、推論処理のプロンプトについて確認する。推論処理とは、算術的な計算や何らかの形で推論が必要な問題を解かせる用途で使用される。例えば次のような問題である（資料 8）。

資料 8　推論タスクの例

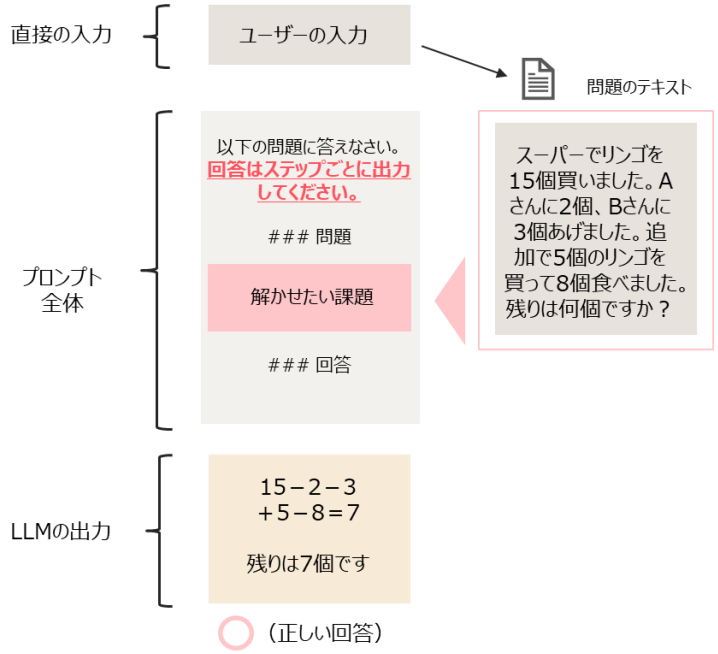


（出所）筆者作成

一般的に、推論処理は LLM にとって最も難しい用途の 1 つであると言われている。これは LLM は単に統計モデルとして確率に従って言葉を選択しているに過ぎず、計算に必要な計算機や専用のプログラムを持っているわけではないことに起因する。しかし、推論ができればより複雑な用途での活用が可能となり、期待されている領域でもある。現在の LLM の性能では、推論処理を十分に実行することが難しく、より高度な PE スキルが必要となっている。

推論処理で特に役立つとされている PE が、chain-of-thought（CoT）プロンプティングと呼ばれる手法である（資料 9）。CoT は、例えるなら小学生に「途中式を省略しないで解いてみよう」と指示を出すようなイメージである。直接答えだけを求めるとミスしがちになるが、中間結果である途中式も一緒に出力させることでミスを減らすことが可能となる。

資料 9　CoT プロンプティングの例



（出所）筆者作成

ここまで、いくつかの活用事例を具体例と共に確認した。今回例示した活用事例も LLM で期待されているタスクのほんの一部であり、プロンプト次第でまだまだより多くの領域へと適用が可能となる。さらに、連携する外部システムや使いどころを上手に選択することで、その可能性は限りなく広がっていく。

7.　おわりに

今後も LLM は間違いなくさらなる進化を遂げていく。しかし、どれだけ性能が良くなろうとも、そのポテンシャルを十二分に発揮できるかは利用者の指示に掛かっている。LLM の進化に置いて行かれないように、利用する側においても日々使いこなしていくための知識やスキルを磨いておくことが重要となる。また、「プロンプトに何の情報を入れてどの様に記述するとどんな結果になるのか」や「どのような手順を踏めば目的を達成できるようになるのか」などの感覚は、兎にも角にも使ってみることで磨かれる。様々な用途や状況でいろいろと試行錯誤し、是非最適なプロンプトについての理解を深めていただきたい。これは、アプリ開発者については言うまでもなく、アプリ利用者についても知っておいて損はないものであろう。そうすることで、実現できる範囲が広がり、想像もしなかった新たな活用方法についても創造していけるようになる。

一方で、アプリ開発者としては、「いかにアプリ利用者にプロンプトを意識させずに

LLM によって生成される価値を提供できるか」についても考えていく必要がある。また、利用者が入力してくるプロンプト次第ではリスクや安全上の問題が生じてしまう可能性があることも忘れてはならない。これらについては、プロンプトデザイン（PD：Prompt Design）（注 2）や敵対的プロンプト（Adversarial Prompting）（注 3）と呼ばれる領域で研究されており、LLM の活用においてどちらも重要な観点となるため、合わせて理解を深めておくと良いだろう。

ChatGPT の登場で世界的に LLM の活用検討が急がれる中、利用者及びアプリ開発者として押さえておくべき「プロンプトエンジニアリング」の基礎について解説した。LLM の活用を考える上で、AI の性能を十分に引き出し達成したい目的を果たせるかは利用する側の指示に掛かっている。プロンプトエンジニアリングによって、どこまで LLM の活用領域が広がっていくのか今後の動向に注目したい。

以上

【注釈】

1) 処理を実行する上で参考となるいくつかの例示をプロンプトに記述して、モデルをより高い性能に導くテクニック。テスト直前に類似の過去問をいくつか解説し、解答方法や回答形式を覚えさせるイメージに近い。
2) 生成 AI を組み込んだアプリ開発において、アプリ利用者の顧客体験やジャーニーに沿ってプロンプトを最適化していくアプローチまたはプロセスのこと。単にアプリ開発者がプロンプトエンジニアリングで LLM を個別の目的にカスタマイズする過程を含め、目的達成に必要な情報の取得から精度向上のための加工、プロンプトへの配置といったプロンプト作成の一部或いは全体に関する最適化への取り組み。
3) プロンプトによる LLM への攻撃手法。悪意を持った特殊なプロンプトを設計してアクセスすることで、不正アクセスやサービス不能などを引き起こす方法。

【参考文献】

- Abulhair Saparov (2022) “Language Models Are Greedy Reasoners: A Systematic Formal Analysis of Chain-of-Thought”
  https://doi.org/10.48550/arXiv.2210.01240
- DAIR.AI (2023) “Prompt Engineering Guide”
  https://www.promptingguide.ai/
- Daixuan Cheng (2023) “UPRISE: Universal Prompt Retrieval for Improving Zero-Shot Evaluation”
  https://doi.org/10.48550/arXiv.2303.08518
- Jiacheng Ye (2023) “Compositional Exemplars for In-context Learning”
  https://doi.org/10.48550/arXiv.2302.05698
- Pengfei Liu (2021) “Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing”
  https://arxiv.org/abs/2107.13586
- Qingxiu Dong (2022) “A Survey on In-context Learning”
  https://doi.org/10.48550/arXiv.2301.00234
- Seonghyeon Ye (2023) “In-Context Instruction Learning”
  https://doi.org/10.48550/arXiv.2302.14691
- Shizhe Diao (2023) “Active Prompting with Chain-of-Thought for Large Language Models”
  https://doi.org/10.48550/arXiv.2302.12246
- Simeng Sun (2023) “How Does In-Context Learning Help Prompt Tuning?”
  https://doi.org/10.48550/arXiv.2302.11521