

LLMを育てるプロンプト・エンジニアリングの基礎知識

北爪 裕紀

表1 プロンプト設計の代表的な手法

観 点	手 法	説 明	具体例	メリット	デメリット
例の有無	Zero-shot プロンプト	例を与えずに指示だけを出す	「この文章を200字以内で要約してください」	手軽で即時に試せる、探索の出発点になる	出力が安定しない、重点や形式がずれる
	Few-shot プロンプト	入出力の「お手本」を示す	「次のように要約してください：例1…例2…」	出力の粒度・表現を安定化できる	例を準備する手間がある
考える手順を指示する	Chain of Thought プロンプト	思考過程を言語化させる	「ステップごとに考えを示してから答えてください」	複雑な推論で精度と説明性が向上する	出力が長くなる、誤推論ももっともらしく見える
出力方法を指示する	構造化出力 プロンプト	出力形式を指定する	「結果をJSON形式で返してください」	業務での後処理が容易になる	指定が曖昧だと崩れやすい

第2部では、LLMを自分専用にするワザとして、LLMへの入力をレベルアップさせる人間側の工夫を紹介します。具体的には、プロンプト設計、API、ハルシネーション抑制のためのデータ整理などを解説します。

(編集部)

LLMを上手く使うには…人間側(プロンプト)の工夫も効果てきめん

生成AIの代表例として、大規模言語モデル(LLM：Large Language Model)が注目を集めています。対話形式で質問や依頼を入力すると、自然な文章で答えや提案を返してくれるため、調査、文書作成、コード生成など、さまざまな業務で活用が進んでいます。しかし、実際に試してみると、思ったような答えが出ない、精度が安定しないと感じることも少なくありません。その理由の多くは、AIの性能だけではなく、AIへの指示(プロンプト)の仕方にあります。

LLMは一見人間の文章を理解しているように見えますが、実際には、ざっくり言えば次に来る単語を予測するという統計的な仕組みで動いています。そのため、プロンプトが曖昧であれば、生成される出力も曖昧になりやすいのです。従って、意図とずれた結果を避けるためにはAIが解釈しやすい形で指示(プロンプト)を設計する必要があります。

そこで本稿では、プロンプト設計の基本手法を紹介します(表1)。シンプルなプロンプトから始め、業務に合わせて育てていく流れを学ぶことで、自分に合っ

たLLMを作り上げるきっかけとなるでしょう。

プロンプト設計には定石がある

LLMへの指示文をプロンプトと呼びます。プロンプトとは、LLMに入力する文章や条件指定の総称であり、質問文、背景説明、出力形式の指定なども含まれます。このプロンプトの内容や構造を、狙い通りの出力を引き出せるように設計する取り組みを、プロンプト・エンジニアリングと呼びます。さまざまな考え方がありますが、表1の大きく3つの観点到に整理できます。

プロンプトの工夫は今すぐ誰でも試せます。加えて適切な指示の出し方はある程度の定石があり、それを知っているかどうかで結果の質が大きく変わります。例えば

この文章を要約してください

という単純な指示では、粒度や重点が期待とずれることがあります。しかし、

この文章を3項目の箇条書きにして、重要な数字は残してください

と具体的に書くと、結果が大きく変わることはイメージしやすいでしょう。

本稿では例としてChatGPTとのやり取りを用いますが、ここで紹介するプロンプト設計の考え方は、多くのLLMに共通して適用できます。なお、本記事に示すプロンプト例はあくまで参考であり、モデルの