TaPrompt：タップ操作による構造化プロンプト修正システム

林 一帆[1]　宮下 芳明[1]

概要：生成 AI を使いこなすには質の高いプロンプトが必要とされており，それを自動生成する技術を用いても細かな意図を反映できないことが多く，何度も文字修正が必要である．本稿では，効率的にプロンプトを修正するためのインタフェースを提案する．目的文，役割指定，コンテキスト指定，出力形式指定，推論ステップという観点において，選択肢を表示し，ユーザーはタップ操作のみで，目的の生成結果を迅速に得られることが期待できる．

1. はじめに

大規模言語モデル（LLM）をはじめとする生成 AI の社会実装が進む一方，その性能はユーザーが入力するプロンプトの質に大きく左右されるという課題が顕在化している．生成 AI から意図に沿った質の高い応答を引き出すには，より詳細かつ具体的なプロンプトが求められるが，多くのユーザー，特に非専門家にとってその設計は容易ではない．

この設計上の課題の根底には，ユーザーの AI に関する背景知識によって LLM に対する認識や期待が大きく異なるという問題が存在する [1]．このような背景知識に起因する認識のずれが原因で，多くのユーザーはプロンプトをうまく設計できず，結果として期待通りの応答を得られずにいる．Liu ら [1] は，プロンプト設計におけるユーザーが陥りがちな失敗パターンとして，以下の 5 つを示した．

(1) **不適切なメンタルモデル:** LLM を人のように捉え，「文脈を察してくれるはず」と過度に期待してしまう．
(2) **不十分な具体性:** 望ましい答えを得るために必要な背景・制約条件・出力形式といった情報が不足している．
(3) **場当たり的な試行錯誤:** 一貫した戦略がなく，思いつきでプロンプトを修正するため非効率に終わる
(4) **指示の不完全な伝達:** 一度に多くのことを頼もうとして，指示が不完全に終わったり，伝わりにくくなったりする．
(5) **過信と早すぎる満足:** 生成された文章が部分的に正しかったり，流暢だと，内容の細かな誤りを見落とし，「完成した」と判断してしまう

このようなユーザーと LLM の間のインタラクション課題に対し，ユーザーの簡単な指示から効果的なプロンプトを自動生成する手法（Automatic Prompt Engineering, APE）[2] が提案されている．しかし，自動化手法もまた LLM に依存するため，本質的な限界を抱えている．すなわち，ユーザーからの明示的な指定がない限り，LLM は一般的・平均的な応答を生成する傾向があり，ユーザー固有の暗黙的な意図や専門的な文脈を完全に反映したプロンプトを生成するには至っていない．結果として，ユーザーは意図した出力を得るために依然として複数回にわたるプロンプト修正を強いられ，APE が本来持つ利便性を損なう一因となっている．

本稿では，これらの課題を解決するため，対話的なプロンプト修正システム **TaPrompt** を提案する．提案システムのメイン画面を図 1 に示す．本システムはユーザーの簡単な指示から効果的なプロンプトを生成するという点においては，APE の一種である．しかし，TaPrompt の特徴としては，従来の APE が単一のテキストブロックとしてプロンプトを出力するのに対し，自動生成されたプロンプトを「機能的構成要素」へと分解し推奨される要素をあらかじめ選択した状態でユーザーに提示する．ユーザーは，提示された要素と出力に応じて選択肢を直感的に選択・修正する対話的なプロセスを通じて，自身の意図を正確に反映したプロンプトを効率的に洗練させることが可能となる．本稿では，TaPrompt の設計と実装を示すとともに，使用例を通して，本システムが非専門家による質の高いプロンプトの効率的な構築に有効であることを実証する．

[1] 明治大学

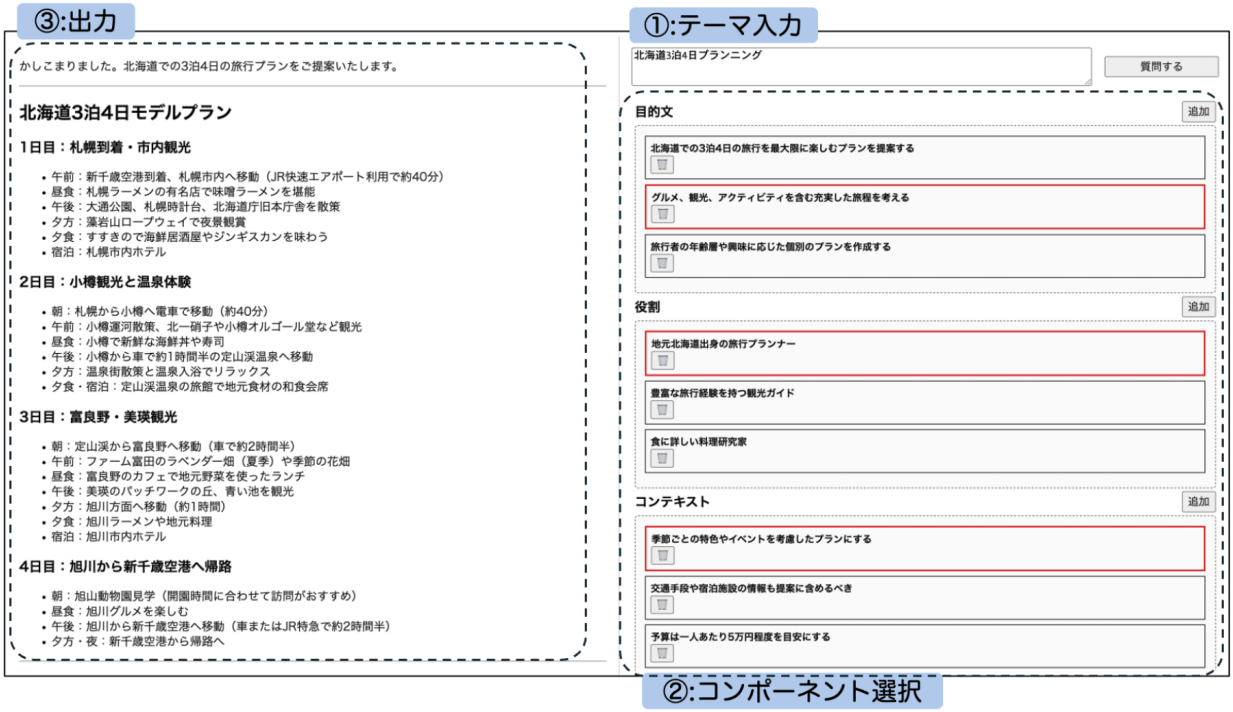

図 1　TaPrompt システムのメイン画面

2. 関連研究

2.1 コンポーネントベースのプロンプトの設計

近年のプロンプトエンジニアリング研究では，単一の指示文よりも「役割・文脈・タスク」など複数の機能的構成要素（コンポーネント）に分解して設計することが推奨されている．Chen ら [3] はこうした構造化により出力の一貫性と信頼性が向上すると報告し，プロンプト設計を「入力の構造化を通じてモデル性能を最大化するプロセス」と位置づけた．

さらに，Lin らは応用言語学領域を対象に，「ペルソナ，対象者，文脈，指示，出力仕様」という五つの要素を列挙し，それぞれを明示的に記述することで専門的な高品質回答が得られると示した．同様に，Bsharat ら [4] もタスク目的・読者設定を明確化し，###Instruction### など書式レベルで指示と文脈を分離することが品質向上に寄与すると述べている．

2.2 プロンプトの作成・改良支援

2.2.1 APE，プロンプトミドルウェアによる支援

プロンプト作成そのものを自動化するアプローチとして，Zhou らによる Automatic Prompt Engineering(APE)[2] が挙げられる．APE(AutoPromptEngineering) は，タスクを記述した少数の入出力例を生成 AI に与えることによって，そのタスクを遂行するのに最適な指示を LLM 自身に生成させる手法である．これにより，人間が試行錯誤するコストを削減する可能性が示された．しかし，このような全自動のアプローチでは，生成されたプロンプトがユーザーの持つ細かな意図や暗黙的な制約を完全に反映できないという課題がある．また，生成プロセスがブラックボックス化しやすく，ユーザーが自身の意図に合わせてプロンプトをさらに改良することが難しい場合がある．

このようなシステムの完全自動化とユーザーの意図反映との間のトレードオフを解消するため，ユーザーインターフェース（UI）を介したアプローチが存在する．その代表的なアプローチの一つが，MacNil らが提唱する「プロンプトミドルウェア」である [5]．これは，UI 上の選択肢や操作を，LLM への具体的なプロンプトにマッピングするための中間層として機能するフレームワークである．プロンプトミドルウェアは，プロンプト生成を支援するという目的において APE と共通するが，ユーザーとのインタラクションを重視する点で特徴がある．

ユーザーによるプロンプトの改良を支援するアプローチは，入力と出力の双方の観点から研究されている．これまでの入力段階での支援とは別に，生成された出力の評価を体系化することで，間接的に改良を促す『出力支援』のアプローチも存在する．例えば，Yang らが提案した EvalLM[6] は，生成 AI を用いて解答候補の品質を自動でスコアリングし，客観的なフィードバックを提供する．

2.2.2 UI コントロールにおける静的・動的アプローチ

ミドルウェアにおける UI 設計は，ユーザーの制御感や認知的な負荷に影響を与える重要な要素である．Drosos らは，ユーザーが求める制御のあり方が画一的ではないこ

とに着目し，静的な UI と動的な UI のトレードオフを検証した．彼らは，事前に定義された汎用的な選択肢を提示する静的プロンプト改善コントロール（Static PRC）と，ユーザーの入力文脈に応じて LLM がその場で関連する選択肢を UI として生成する動的プロンプト改善コントロール（Dynamic PRC）を比較した．その結果，適応的な UI（動的）は，ユーザーにより多くの制御感を与え，タスクの探求を促す点で好まれる一方，生成される選択肢の効果を推論することが難しく，認知的な負荷を高める可能性があるというトレードオフが示された．

2.3 本研究の位置付け

先行研究では，プロンプトの改良プロセスを支援するため，主に二つの方向性からアプローチがなされてきた．一つは，Drosos らが示した動的プロンプト改善コントロール（Dynamic PRC）[7] のように，UI を介してユーザーの入力操作をガイドし，修正の「方針」を提示する入力支援である．もう一つは，Yang らが提案した EvalLM[6] のように，生成 AI を用いて出力結果を評価し，改良のための「フィードバック」を提供する出力支援である．

これらの手法はプロンプト改良の対話化に大きく貢献した一方で，依然として課題も残されている．入力支援アプローチでは，具体的な修正文言の考案は利用者に委ねられており，依然として認知的な負荷を伴う．本研究では，システムによってコンポーネントごとに提示される選択肢を「タップ」というインタラクションを通じて選択することで，プロンプトを洗練・修正するプロセスを簡易化し，この課題にアプローチする．

3. 提案手法

本システムの動作画面を図 1 に示す．本システムでは，ユーザが対話的な操作を通じて効率的にプロンプトを構築・修正できる，新たなインタフェース **TaPrompt** を提案する．本システムでは，生成 AI を用いてプロンプトを 5 つの構成要素（目的・役割・コンテキスト・出力形式・推論ステップ）に分解してキーワードを生成し，それぞれの要素について複数の候補をユーザーに提示する．ユーザーはこれらの候補を組み合わせ，最終的な出力をリアルタイムで確認しながら，自身の意図に沿ったプロンプトを効率的に構築することが可能である．本システムにおいて，OpenAI[8] の API を使用し，モデルは gpt-4.1-mini-2025-04-14 である．

3.1 設計思想：コンポーネントベース・プロンプティング

良質な出力を得るためには，指示の明確さや背景情報の提供が重要であると考えられる．そこで本手法では，先行研究 [9] を参考に，単一のテキストとして扱われがちなプロンプトを以下の 5 つのコンポーネントに分解する．

(1) **目的：**プロンプト全体の核となる部分で，ユーザーが最終的に何を得たいのかというゴールを明確に言語化する．生成 AI は目的からタスクの全体像を把握し，後続の指示との一貫性を保ちながら，より正確な回答を生成しようとする．

(2) **役割：**生成 AI に特定の専門家やキャラクターなどの「役割（ペルソナ）」を与える．生成 AI がどのような立場・専門性を帯びて応答すべきかを宣言する．

(3) **コンテキスト：**生成 AI の回答を生成するために必要な背景情報，前提条件，制約などを与える部分である．さらに，ユーザーの指示を明確化することが可能である．また，生成 AI は一般的な回答ではなく，ユーザーの特定の状況に即した回答を出力することが可能になっている．

(4) **出力形式：**生成 AI に生成してほしいアウトプット形式や構造を具体的に指示する．例えば，「JSON 形式で出力」や「マークダウンの箇条書きでまとめる」などの指示が挙げられる．これにより，生成された回答を人間が読みやすい形に整えたりする手間を削減することができる．

(5) **推論ステップ：**複雑な問題や論理的思考が求められるタスクにおいて有効な要素．モデルの思考の過程を外化させ，整合性を確保できる．

3.2 システム概要と処理フロー

提案手法を実装したシステムの概要と処理フローを図 1 に示す．本システムは，ユーザーが対話的にプロンプトを構築・修正し，その結果をリアルタイムで確認できるインタフェースを提供する．処理は以下のステップで進行する．

step1：テーマ入力

まず，ユーザーは UI 上部の入力領域（図 1-1）に，たとえば「北海道 3 泊 4 日のプランニング」といったテーマを入力する．

step2：プロンプト要素の自動生成と初期提示

テーマが入力されると，システムが生成 AI を用い，3.1 節で定義した 5 つのコンポーネントそれぞれについて，テーマに沿ったプロンプト要素の候補を 3 つずつ自動生成する．同時に，システムは最も適切と判断した要素の組み合わせを初期プロンプトとして自動選択し，その結果生成される生成 AI の出力（図 1-3）を提示する．

step3：ユーザーによる対話的な調整

ユーザーは，提示された出力を確認し，意図と異なる場合は，UI 中央のプロンプト要素選択エリア（図 1-2）で各コンポーネントの選択肢を変更する．選択が変更されるたび，最終的なプロンプトとそれに応じたの出力（図 1-3）は即座に更新される．ユーザーは，このインタラクティブなプロセスを通じて，自身の要求に合致した出力を効率的に探索することが可能となる．

目的文	北海道の観光名所を含む3泊4日の旅行プランを提案する 地元の食材を活かした美食体験を重視したプランを作成する 家族向けのアクティビティを組み入れた楽しい旅行日程を考える
役割	北海道の観光業に精通した旅行プランナー 地元の文化や歴史に詳しいガイド アクティブなアウトドア愛好者
コンテキスト	総予算は10万円以内で、移動手段はレンタカーを予定している ターゲットは小さな子供を持つ家族で、アクティビティはバランスを重視する 訪れる時期は夏休みで、混雑具合に注意する必要がある
出力形式	日ごとのスケジュールをまとめた表形式で出力する 各日ごとの詳細なアクティビティと食事内容を説明する文書形式で出力する 短い箇条書き形式で、各アクティビティの時間と場所を明示する
推論ステップ	各観光スポットの所要時間や移動距離を考慮して日程を組むこと 口コミや評価を基に、食事場所を選定するプロセスを含む 天候や特別イベントを考慮して、柔軟にプランを調整すること

図 2　プランニングタスクに対する生成プロンプト

北海道3泊4日夏休み旅行プランをご提案いたします。混雑を考慮しつつ、効率よく主要な観光名所を巡る内容です。

【1日目】札幌到着・市内観光とグルメ体験
午前：新千歳空港到着後、快速エアポートで約40分、札幌駅へ移動
札幌駅周辺で荷物を預けたら、徒歩または地下鉄で「大通公園」へ。広大な都会のオアシスを散策しながら季節の花やイベントを楽しみます（所要約1時間）。
昼食は「札幌ラーメン共和国」内で北海道味噌ラーメンを味わい、混雑を避けるため早めの時間帯に入店がおすすめです。
午後：「札幌時計台」見学と「北海道庁旧本庁舎」も徒歩圏内。歴史的建造物を巡りつつ、写真撮影などでゆっくり過ごします（所要1.5時間）。
その後、市電で「藻岩山」へ移動。山頂からの夜景は「ミシュラン・グリーンガイド・ジャポン」で2つ星を獲得しているほどの絶景です。混雑緩和のため、ロープウェイのチケットは事前購入を推奨します。
夕食は藻岩山近くのレストランで地元食材を活かしたコース料理をお楽しみください。

【2日目】小樽日帰り観光と海鮮グルメ
朝：札幌駅からJR快速で約30分の小樽へ。夏休みの混雑を避けるため、早めの出発を心がけます。
午前：小樽運河沿いや堺町通りを散策。歴史的な倉庫群、ガラス工房やオルゴール館を見学しながらゆったりとショッピングや体験を楽しみます（所要約3時間）。
昼食は小樽の海鮮丼が人気の店で旬の魚介を味わってください。
午後：「祝津展望台」へバス移動し、日本海の景色を堪能した後、近くの「ニッカウヰスキー北海道蒸溜所」を見学ツアーに参加。蒸留所の歴史やウイスキー製造過程を学べます（所要約2時間）。こちらも混雑するため事前予約が望ましいです。
夕方：札幌に戻り、すすきのエリアでジンギスカン焼肉を夕食に。予約制の有名店利用で待ち時間を抑えましょう。

図 3　図 2 で選択した要素による生成結果

4.　使用例

本章では，**TaPrompt** を用いて二種類のタスクに対するプロンプトの生成および修正を行い，その有効性と修正箇所の妥当性を後述の考察章で論じる．対象タスクは次の二つである．

- **プランニングタスク**：北海道を巡る 3 泊 4 日の旅行プラン
- **プログラム理解タスク**：Python コード（シェルソート）の解説

4.1　使用例 1：プランニングタスク

複数の制約条件（時間・場所・予算・移動手段）を満たしつつ順序立てた旅行プランを自動生成するタスクを設定した．テーマは「北海道の 3 泊 4 日プランニング」とし，**TaPrompt** によりプロンプトを生成した．

4.1.1　生成結果

生成されたプロンプトを図 2 に，ユーザが選択した要素を用いて得られた出力例を図 3 に示す．図中の赤文字はユーザが選択したテキストである．

4.1.2　修正後の結果

修正前の文書形式では観光地に関して詳細に示される反面，時刻や場所が文中に埋もれやすかったため，出力形式を「各日ごとの詳細なアクティビティと食事内容を説明する文書形式」から，「各アクティビティの時間と場所を明示した短い箇条書き形式」へ変更した．修正後の出力を図 4 に示す．

4.1.3　プロンプト変更後の出力適合性

推論ステップで明示した「各観光スポットの所要時間・移動距離を考慮して日程を組むこと」という要件は，変更前（図 3）・変更後（図 4）の双方で満たされていた．具体的には，滞在時間・移動時間が時系列に沿って記載されており，プロンプト指示が忠実に反映されている．さらに，出力形式の指定を「詳細な文章」から「箇条書き」へ変更した際も，フォーマットが即座に切り替わり，所要時間や移動手段が要約された形で提示された．ただし，旅程中の立寄都市は「札幌→小樽→富良野・美瑛→札幌」から「札幌周辺→富良野・美瑛→函館→函館・新千歳」へと部分的に変化しており，副次的な訪問地はモデルによって自律的に再構成されることが確認された．

■1日目（札幌周辺）

- 09:00 新千歳空港着・レンタカーまたはJR移動
- 10:30 札幌市時計台（滞在時間30分）
- 11:15 大通公園散策（滞在時間1時間）
- 12:30 札幌市内でランチ（ジンギスカンなど北海道名物）
- 14:00 札幌市円山動物園（滞在時間1.5時間）
- 16:00 札幌テレビ塔展望台（滞在時間30分）
- 17:00 ホテルチェックイン（札幌中心部）
- 18:30 すすきのエリアで夕食（混雑を避けて早めの入店推奨）

■2日目（富良野・美瑛エリア）

- 07:30 札幌出発、車で富良野へ（所要約2時間）
- 09:30 ファーム富田（ラベンダー畑など、滞在時間1.5時間）
- 11:15 四季彩の丘（広大な花畑、滞在時間1.5時間）
- 13:00 富良野でランチ（地元グルメ推奨）
- 14:30 美瑛パッチワークの路ドライブ・写真撮影スポット巡り（滞在1.5時間）
- 16:30 美瑛町内の宿泊施設チェックイン
- 18:00 夕食（地元食材を活かしたレストラン）

図 4　出力形式変更後の生成結果

目的文	シェルソートアルゴリズムの動作を詳細に説明する コードの各行の機能と目的を分かりやすく解説する 初心者向けにアルゴリズムの背後にある理論も含めて解説する
役割	プログラミング教育に特化した講師 アルゴリズムに精通したソフトウェアエンジニア 初心者向けのプログラミング書籍の著者
コンテキスト	コードに注釈を付けて視覚的に説明する 読者はプログラミング上級者であり、アルゴリズムの使い方や利用例を説明する プログラミング、アルゴリズム初心者に向けて丁寧に説明する
出力形式	段落ごとに分かりやすく解説する形式で出力する 各コードブロックを見出しとして扱い、それぞれの箇所を解説する形式にする 表形式を利用して処理の流れを視覚化する
推論ステップ	まずはコード全体のフローを説明し、その後に関数ごとに詳細を解説する コード内の変数や関数の役割を一つずつ明確にし、それに基づいて順を追って解説すること アルゴリズムの性能についても触れ、その重要性を説明する

図 5　コード解説タスクに対する生成プロンプト

4.2　使用例 2：Python のコード解説

次に，ソースコードの動作を初学者にも理解しやすい自然言語で段階的に説明するプロンプトを **TaPrompt** により生成し，その修正過程を検証した．題材としてシェルソートの Python 実装を用い，ソースコードは付録にて掲載する．

4.2.1　生成結果

生成されたプロンプトを図 5 に，ユーザが選択した要素を用いて得られた出力例の一部を図 6 に示す．図中の赤文字はユーザが選択したテキストである．

4.2.2　修正後の結果

初心者向けの説明では，アルゴリズムの基礎概念や逐次手順が丁寧に解説される一方で，アルゴリズムの応用シナリオについて考えられないため，コンテキストを「プログ

図 6　図 5 で選択した要素による生成結果の一部

図 7　コンテキスト変更後の生成結果

ラミング・アルゴリズム初心者向けの丁寧な説明」から，「読者はプログラミング上級者であり，アルゴリズムの使い方や応用例を中心に説明する」へ変更した．修正後の出力を図 7 に示す．

4.2.3　プロンプト変更後の出力適合性

プロンプトの「コンテキスト」を「初心者向け」から「上級者向け」へ変更した結果，説明文の語彙レベルと深度が適切に調整された．変更前は型ヒントや // 演算子の意味を平易に解説していたのに対し，変更後は「参照透過性」「副作用」など上級者向けの概念を導入し，初歩的な用語解説を省略していることが確認できる．

5.　考察

本章では，4 で実施した 旅行プランニングタスク（図 3-4）および Python コード解説タスク（図 6–7）の生成結果を再検証し，

1：　プロンプト変更への追従性，
2：　同一プロンプトを複数回実行した際の出力ばらつきの傾向
3：　1 で示したプロンプト設計課題への貢献

という 3 つの観点からシステムの特性を分析する．

5.1　プロンプト変更への追従性

2 種類のタスク（旅行プランニングタスク，Python コード解説タスク）に関して，セクション 4.1.3，4.2.3 において示した通り，使用例 1 の旅行プランニングタスクでは出力形式を詳細文から箇条書きへの変更し，一回の生成で忠実に反映されつつ所用時間と移動順序など他プロンプトによる指示も保持されたことが確認できる．さらに，使用例 2 の Python コード解説タスクでは，「プログラミング初心者向け」から「プログラミング上級者向け」へと変更し，これにより出力される解説文の語彙レベルと専門性が大きく向上していることが確認できた．したがって，**TaPrompt** は両タスクでユーザー指定の変更を即時反映できることが観察された．しかし，副次的要素（旅行先等）は LLM が再構成するため，意図外の変更も起こり得ることがわかった．一方で，本システムの試行を通じて，当初意図していなかった選択肢が提示されることで，新たな要求を発見する場面も確認された．このことは，提示された選択肢を比較・検討するプロセスが，ユーザー自身の潜在的な要求の明確化や，曖昧な意図の具体化を支援する可能性を示唆している．

5.2　出力のばらつきに関する分析

5.2.1　使用例 1：旅行プランニングタスク

同一プロンプトを 10 回生成した際，必須要素（日程構成，移動時間，所要時間）は全出力で一貫して含まれていた一方，任意要素（立寄都市，グルメ推奨，細かなアクティビティ）は回答ごとに選択・粒度が揺らいだ．主要都市（札幌，小樽）は高頻度で登場するが，旭川・函館・苫小牧などの登場頻度は均等に分散しやすい傾向がある．グルメ情報についても，「寿司屋通りで海鮮丼」といった具体的推奨から「地元グルメを堪能」といった抽象表現まで振れ幅が大きい．これは，大規模言語モデルが ハード制約（必須条件）は高い忠実度で満たす一方，ソフト制約（補足情報）は多様性を優先する特性に起因すると考える．

5.2.2　使用例 2：Python コード解説タスク

全ての出力で「コードブロック見出し＋解説」という骨格は維持され，初学者向けキーワード補足や処理フロー説明が共通要素として現れた．一方，文体は口語体／論文体，解説の深度は表層説明〜専門的掘り下げまで広く分散した．特に「参照渡し」「計算量解析」などの高度トピックが含まれるか否かは生成ごとにばらついた．これは，モデ

ルが「上級者」という曖昧な属性を多様に解釈し得るためであり，追加の制約（例：具体的に触れる専門項目を列挙）を与えることで分散を収束できると考えられる．

5.3 プロンプト設計課題への貢献

本章では，1 にて示したプロンプト設計における 5 つの失敗パターンに対し，TaPrompt の特性がどのように貢献するのかを論じる．TaPrompt の体系的なアプローチは，プロンプト設計における複数の課題に対処する．まず，プロンプトを「目的」や「コンテキスト」といった機能的要素へ分解する手法は，ユーザーに求められる情報の種類を明確化し，「不十分な具体性」や「指示の不完全な伝達」といった問題の緩和に寄与する．特に，ユーザーからの大まかなテーマ入力に対し，システムが具体的な初期プロンプトを自動生成する機能は，「不適切なメンタルモデル」の課題に有効である．AI との対話に求められる具体性の水準が初期状態で提示され，常に各コンポーネントに選択肢が設定された状態が保証されるため，ユーザーは自然とその基準を学習することができる．

次に，修正プロセスにおいても本システムはユーザーを支援する．「場当たり的な試行錯誤」に対しては，修正の方向性を具体的な選択肢として提示することが有効に機能する．ユーザーは「役割を専門家にする」といった選択肢から出力結果を予測しつつ，意図を持った効率的な修正を進めることが可能となる．さらに，複数の選択肢が常に比較可能な状態で提示されることは，「過信と早すぎる満足」を抑制する上で重要な役割を果たす．たとえ最初の出力が良好に見えても，他の選択肢を試すことで得られる異なる結果に触れる体験は，単一の出力に対する評価をより客観的なものにし，ユーザーの深い探求を促すきっかけとなりうる．

5.4 まとめと今後の課題

プロンプト修正が出力品質に直接反映される一方，**TaPrompt** でハード制約（必須条件）として与えなかった要素は回答の多様性が大きく保たれることがわかった．よりユーザーの意図を細かく反映するためには，想定外のバリエーションを抑制するような UI 設計を行う必要がある．

6. おわりに

本研究では，生成 AI の性能を最大限に引き出す上で多くの利用者が直面する，プロンプト設計の複雑さという課題に着目した．従来の自動プロンプト生成（APE）は，その利便性の一方で利用者の細かな意図を汲み取れず，手作業による修正が不可欠であるという限界を抱えていた．

この課題を解決するため，本稿では対話的なプロンプト修正システム「TaPrompt」を提案した．具体的なタスク（例：プランニングとコード解説タスク）における 2 つの使用例を通じて本システムの有効性を検証した結果，TaPrompt がユーザーの意図を的確に反映した高品質なプロンプトを，効率的に生成可能であることが示唆された．

本研究の貢献は，プロンプトの初期生成から修正に至るプロセスを包括的に支援することで，専門知識のないユーザーでも，自らの意図に沿った出力を容易に得られる環境を実現した点にある．一方で，ユーザーが明示的に指定しなかった要素については，出力に意図しない多様性が生じることも確認された．今後は，この出力の多様性を適切に制御する手法を検討し，よりユーザーの細かな意図を反映するための UI 設計に取り組む．

参考文献

[1] Zamfirescu-Pereira, J., Wong, R. Y., Hartmann, B. and Yang, Q.: Why Johnny Can't Prompt: How Non-AI Experts Try (and Fail) to Design LLM Prompts, *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI '23, New York, NY, USA, Association for Computing Machinery, (online), DOI: 10.1145/3544548.3581388 (2023).

[2] Zhou, Y., Muresanu, A. I., Han, Z., Paster, K., Pitis, S., Chan, H. and Ba, J.: Large Language Models Are Human-Level Prompt Engineers (2023).

[3] Chen, B., Zhang, Z., Langrené, N. and Zhu, S.: Unleashing the potential of prompt engineering for large language models, *Patterns* (2025).

[4] Bsharat, S. M., Myrzakhan, A. and Shen, Z.: Principled instructions are all you need for questioning llama-1/2, gpt-3.5/4, *arXiv preprint arXiv:2312.16171* (2023).

[5] MacNeil, S., Tran, A., Kim, J., Huang, Z., Bernstein, S. and Mogil, D.: Prompt Middleware: Mapping Prompts for Large Language Models to UI Affordances (2023).

[6] Kim, T. S., Lee, Y., Shin, J., Kim, Y.-H. and Kim, J.: EvalLM: Interactive Evaluation of Large Language Model Prompts on User-Defined Criteria, *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, CHI '24, New York, NY, USA, Association for Computing Machinery, (online), DOI: 10.1145/3613904.3642216 (2024).

[7] Drosos, I., Williams, J., Sarkar, A., Wilson, N., Rintel, S. and Panda, P.: Dynamic Prompt Middleware: Contextual Prompt Refinement Controls for Comprehension Tasks, *Proceedings of the 4th Annual Symposium on Human-Computer Interaction for Work*, CHIWORK '25, New York, NY, USA, Association for Computing Machinery, (online), DOI: 10.1145/3729176.3729203 (2025).

[8] OpenAI: Introducing GPT-4.1 in the API, `https://openai.com/index/gpt-4-1/` (2025). Web page, accessed 2025-07-15.

[9] Lin, Z.: Prompt Engineering for Applied Linguistics: Elements, Examples, Techniques, and Strategies, *English Language Teaching*, Vol. 17, pp. 14–14 (online), DOI: 10.5539/elt.v17n9p14 (2024).