情報可視化システム開発における大規模言語モデルプロンプトの定量的調査

平林 晴馬[1]　宮崎 翔[1]　矢谷 浩司[1]

概要：自然言語処理技術に基づく大規模言語モデル (LLM) の進化により，高い生成能力を示す AI 技術がシステム開発やプログラミングの分野にも影響を与えている．本研究では対話型のテキスト生成 AI の利用に関して, 情報可視化システムの開発におけるユーザの行動パターンと利用目的に焦点を当てた定量的な分析を行う情報検索におけるセッション分析の観点を取り入れ，連続して入力されたプロンプト同士の関係性にも注目して AI の利用実態を考察する．さらに，これらの利用実態の調査結果からユーザの生成 AI を用いたシステム開発を支援するツールの設計について考察を行う．分析の結果，ユーザの対話型 AI の利用が AI を主軸としてコードを積極的に記述させる段階と，AI をコード記述の支援に利用してユーザが主体となってコードを記述する 2 つの段階に分類されることが確認された．また，ユーザと AI 間でのシステム開発の進捗状況の同期の難しさが存在することが確認され，支援の必要性が確認された．

A Quantitative Investigation on Use of LLM in Information Visualization System Development

Haruma HIRABAYASHI[1]　Kakeru MIYAZAKI[1]　Koji YATANI[1]

1. はじめに

自然言語処理技術に基づく大規模言語モデル（LLM）の飛躍的な進化は，AI 技術の領域において革新的な発展をもたらしている [21]．これらの先進的なモデルは，ユーザの入力に対してより自然で洗練された応答を提供し [23]，同時に創造的な画像生成などの複雑なタスクを高い精度で実行する能力を実現している [22, 25]．

プログラミングやソフトウェア開発の分野においても，このような生成 AI の技術が大きな影響を及ぼしており，これまでニューラル・モデルなどの機械学習技術によって研究されてきたコードの自動生成 [26] や, コードのエラー修正 [17] などの点においてこれまでにない効率と精度を実現している [5, 6, 20]．

このような生成 AI の利用拡大に伴い，ヒューマン・コンピュータ・インタラクション (HCI) の分野では，人間 (ユーザ) の生成 AI の利用に関して，その有効性や実態を調査する研究が行われている．このような研究の中には，生成 AI が不正確な情報を提供したり [4]，作業の効率を低下させる例 [1] を報告するものも存在し，生成 AI の有効な利用手法の検討が必要となっている [14]．このような背景から，生成 AI の利用実態をもとに，適切な利用を調査する研究が活発に行われており，特に，ユーザの生成 AI に対する入力 (プロンプト) を軸としたプロンプト・エンジニアリングでは，自然言語や画像生成などの様々な用途で生成 AI の応答の精度を高めるプロンプトの入力方法や利用戦略，支援手法が研究されている [3, 11, 13, 23]．

本研究では，コーディングの実施にとどまらず，システムの実装方針の検討段階などを含めた包括的なデータ可視化システムの開発プロトタイピングを調査対象とし，プロンプト・エンジニアリングの観点からプロンプト・データを分類・分析し，ユーザの利用実態を深く掘り下げる．実際，このような生成 AI を用いたコード作成の利用実態やコード支援の可能性の調査はこれまでにも行われているものの，その多くが特定のアルゴリズムや簡易的なシステムを作成するコーディング課題の実施の観察 [2, 8, 10] や，定性的な調査としての利用者へのインタビューを行う形

[1] 東京大学 Interactive Intelligent Systems Laboratory

式 [2,9,15,18] をとっており，より自由なシステム開発の観察や実際のプロンプトなどの利用データに基づく定量的な視点での分析は行われていない．今後生成 AI の利用が拡大する中でシステム開発において生成 AI がどのように利用されうるかを考察し，支援の可能性を検討するにあたっては単純なコード記述における生成 AI の利用にとどまらずシステム開発全体を通した利用の観察が必要であると考えられる．また，これまで行われてきた定性的考察に定量的な論拠を示すことでより客観性の高い考察を行うためにデータに基づく量的視点を導入する．

本研究では，東京大学の学部生向けに開講される「電気電子情報実験・演習第二: 情報可視化とデータ解析」講義において，Web 上で動作する大規模なデータを可視化するシステムの作成における GPT-4 のモデルに基づく ChatBot システムの利用を行ってもらい，プロンプトデータを収集した．そこで取得した実験参加者のプロンプトデータの分析をもとに，オープンコーディングによるプロンプトの目的別分類を実施した．また，生成 AI に対するプロンプトと情報検索の分野における Web 検索時の入力 (クエリ) の類似性から，クエリ分析を参考にしたセッション[*1]ごとのプロンプトの分類を行い，その関係性を調査した．このような分析からシステム開発における生成 AI の利用実態の調査と，その支援手法の考察を行った．

本研究の研究的観点としては，

1. システム開発においてユーザがどのような目的で対話型 AI を利用するか
2. 対話型 AI を用いたシステムの開発におけるセッションの観点の導入によりどのような傾向が見られるか
3. システムの開発における AI 支援はどのように行われるべきか

となっており，これらの観点から利用者の生成 AI の利用を観察し，その実態と支援の手法について考察する．

2. 関連研究

2.1 プロンプト・エンジニアリング

プロンプト・エンジニアリングの分野では基礎研究として，生成 AI に入力されるプロンプトの分類が行われており，White ら [23,24] は一般的な生成 AI の利用とコーディング時の生成 AI の利用においてインターネット上で公開されているプロンプト入力のデータセットを観察し，目的に応じた分類を行っている．特にコーディングに関するプロンプト入力では 4 つの大きなカテゴリと 14 つの小さなカテゴリに分類され，さらにそれぞれの分類のプロンプトがどのように組み合わされ，利用されるかを紹介している．また，Liu ら [13] は生成 AI に対して不適切な言動や出力を誘発する jail breaking prompt (脱獄プロンプト) と呼ばれるプロンプトに関して研究を行っている．この研究ではオープンコーディング [19] の手法を用いて脱獄プロンプトの分類を行っており，分類した群ごとに生成 AI の応答に関して比較・分析を行っている．

入力に対する研究としてはテキストベースの入力だけでなく，Myers ら [16] は自然言語処理を用いた音声インタフェース・システムの利用実態を調査し，利用上の問題が生じた際に利用者がどのような手段で問題を解決したかを分類し，手段ごとの共起性などの関連性を分析した．この分析から利用者の傾向を考察し，利用者の問題解決を支援する機能の可能性を報告している．

プロンプトの分類・分析に類似する研究分野として，情報検索の分野で行われるクエリ分析が挙げられる，クエリ分析の分野では Web 検索において目的の情報を取得するまでに入力されるクエリをセッションごとに分割し，セッション内のクエリやセッション同士の関係性を分析している [7]．Xie ら [25] が画像生成におけるプロンプトの分類と分析を行った研究では，そのクエリ分析との類似性を指摘している一方で，プロンプト分析においてはクエリ分析のセッションよりも複雑な構造の中でユーザの意図や戦略などの分類が必要とされることを述べている．

2.2 コーディングにおける生成 AI の利用

実際，生成 AI がどのようにコーディング時に利用されるかをプロンプト・エンジニアリングの視点で調査した研究として，Kazemitabaar [10] らは，初学者による生成 AI を用いた Python のコーディング課題の実施を観察し，プロンプト入力時の状況や内容の分類と内訳を示した．また，生成 AI の利用戦略を 4 つに分類し，戦略と課題実施後の学習効果の関係性から生成 AI を頻繁に利用したり，全く利用しなかった場合に比べて，生成 AI によるコード生成とユーザ自身によるコード記述を統合的に行った場合の方が Python の記法に関する学習効果が高かったことが報告されており，生成 AI とユーザが共同でプログラミングを行うことを促進する支援デザインの可能性を示唆している．また，Jiang ら [8] は HTML と JavaScript を用いた Web アプリケーションの作成においてコードの中に入力されたプロンプトからコードを生成する独自の LLM システム GenLine を作成し，その利用の観察とデザイン改善の考察を行っている．利用の観察ではコーディング経験の有無によって入力手法に異なる傾向があることを報告している．またシステムの出力の精度を出力の採用率によって数値化し，入力手法と精度の関係性を分析することで，システムの出力精度を向上させるプロンプトへの変換機能の可能性を示した．ただ，これらの研究では生成 AI の利用場面として単純なコーディング課題の実施に留まっており，またプロンプトの分類・分析として 2.1 で述べた Myers ら [16] が音声インタフェースシステムに対して行ったようなプロ

[*1] 目的の情報を取得するまでに入力されるクエリのまとまり

ンプト同士の関連性の分析は行われていない．

利用実態の調査としてはインタビューによる質的調査も行われており，Barke ら [2] は，GitHub Copilot を用いたコーディング課題を実施し，実施後のインタビューから利用実態の質的分析を行っている．その中で，生成 AI を利用したコーディングでは生成 AI を用いながらも利用者が積極的にコードを書いていく accelerate mode と，現状の解決策が明らかになっていない状態で生成 AI にコードを記述させていく exploration mode の 2 種類の利用形態が存在することを報告しており，実際に利用者がタスク中にどのようにこの 2 形態の利用を往来するか定量的観点で示している．また，コード支援に関する考察では，これらの利用形態を入力内容から判別することでユーザが求める支援を提供することが重要であると考察している．

このようなコーディングにおける AI 技術の利用に関してはこれまで単純なコーディング課題の実施やインタビューによる定性的な評価が主となっており，実際のシステム開発のプロトタイピングを長期的に観察・調査した研究やデータに基づく定量的な分析は多く行われていない．そこで本研究では単純なコード記述に留まらず，システムの設計や構想段階も含めた包括的な観察と取得されたプロンプトデータの定量的分析からシステム開発における生成 AI の利用の実態とその支援の方向性を考察する．

3. データの収集

3.1 情報可視化実験の概要

本研究では，対話型 AI を用いたシステム開発におけるプロンプトデータを収集するため，東京大学の学生向けに行われる講義「電気電子情報実験・演習第二: 情報可視化とデータ解析」(以下，情報可視化実験) においてデータ収集を実施した．情報可視化実験は三ヶ月ほどの期間の中で十日に渡って実施され，参加生徒は二人ないし三人のグループで HTML や JavaScript などを用いてブラウザ上で動作する情報可視化システムの制作を行う．データの可視化には統一して D3.js[*2] と呼ばれる JavaScript の情報可視化に特化したライブラリが用いられる．

また，本研究のデータ収集のために OpenAI が発行する GPT モデルの API キーを利用しながら ChatGPT の UI による操作を可能にする Chatbot UI[*3] を本研究用に改良した ChatBot を利用してもらった．

3.2 研究参加への同意確認

講義に参加する生徒に対して研究参加の同意確認を行い，44 人から本研究への参加の同意が得られた．

*2 https://d3js.org/
*3 https://www.chatbotui.com/ja

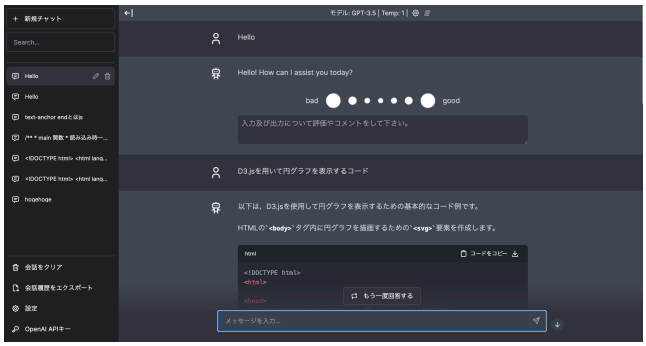

図 1: ChatBot システムの UI．ChatGPT の UI を模した仕様となっている．

表 1: 実験参加者の Web 関連のシステム実装経験に関する構成．

HTML や JavaScript に対する事前知識	人数
Web 関連は全く触ったことがない	13
HTML くらいは触ったことがあるが，JavaScript はない	15
多少触ったことがあり，簡単な Web サービスなら実装できる	8

3.3 情報可視化実験内での生成 AI の利用とデータの収集

講義内では，実験に関わる全てのタスクにおいて生成 AI の利用を可能とし，積極的な利用を促した．講義最終回終了後，講義中に利用した ChatBot システム内の全てのログデータを提出してもらった．本研究への参加を同意した参加者の中には実験期間中にこちらが提供した ChatBot システムを利用しなかった人や，個人で別のシステムを利用した人が数名おり，全員からデータを収集するには至らず，最終的に 36 人 (表 1) から 5912 件のプロンプトと生成 AI の応答のデータを収集できた．

3.4 講義最終回後のアンケート調査の実施

講義の最終回後には，情報可視化システムの実装において利用した ChatBot の利用目的や，詳細な利用の実態について問うアンケートを行った．本研究ではデータに基づいた対話型 AI の利用を分析していくが，得られた結果に関して，アンケートへの回答と照らし合わせ，検証を行っていく．

4. データの選別・分類

本節では取得したプロンプト・データの分析対象の選別と分類の実施について記述する．

4.1 分析対象データの選別

データの収集段階では ChatBot システム内の全てのデータを提出してもらった．収集したデータの中には実験前半

図 2: データの分類のために作成した分類用のシステム画面．左側に前後のデータを表示し，関連性を確かめながら分類を行った．

で行われるチュートリアル課題に取り組むために利用されたものや実験と関連性が低いと見られるデータが含まれていた．本研究ではプログラミングによるシステム開発における生成 AI の利用方法を調査するため，実験内で行われるチュートリアルなどに関わるプロンプトデータなどは分析対象外とし，最終的に制作する情報可視化システムに関するやりとりのみを分析対象データとした．その結果，4400 件のデータが分析対象として選定された．

4.2 データの分類

データの分類においてはプロンプト入力の文脈を理解し，正確に分類を行うため，前後のプロンプト同士の関連性を加味しながら分類を行っていく必要がある．そこで，図 2 に示すような分類用のアプリケーションを作成し，画面内で前後のデータを確認しながら，その関連性を加味して分類を行った．分類観点としては，以下に示す二つの観点で分類を行った．

- セッション単位での初期プロンプトと後続プロンプトの分類
- 入力目的別の分類

それぞれの分類に関する説明と分類の結果をまとめる．

4.2.1 入力プロンプトのセッションごとの分類

情報検索の分野におけるセッションの概念をプロンプト分析にも導入し，分類を行った．情報検索の分野では目的の情報を取得するまでに入力されるクエリ群をセッションと定義しており，これをプロンプト分析にも導入するため，jansen ら [7] が行ったようにプロンプトをセッションの開始に入力される「初期プロンプト」とその後に続く「後続プロンプト」に分類を行った．二つの分類の定義に関しては 4.2.2 内で後述する分類項目の検討段階で適切な定義を検討し，それぞれ以下のように設定した．

- 初期プロンプト … 以前のやりとりを参照せず，独立した目的を持った入力
- 後続プロンプト … 以前のやりとりを参照しており，同一または関連した目的を持った入力

以上の分類をもとに「初期プロンプト」とそれに対応する「後続プロンプト」を一つのセッションとして定義した．

表 2: セッションを構成する初期プロンプトと後続プロンプトの分類結果．

分類	出現回数
初期プロンプト	856
後続プロンプト	3544

表 3: 実験参加者によって入力されたプロンプトの目的別分類．複数ラベリングを許容し，前後のプロンプトの関係性も加味して分類を行った．

分類	定義	回数
アイディア創出	複数の選択肢の列挙の要求	66
システム設計の構想・実装方針の検討	システム作成の方針の検討	212
注目対象の共有・現状の同期	コードの添付や視覚的情報など現状の伝達	1755
コード生成要求	コードの仕様伝達と生成の要求	1849
出力の改善要求	要件理解・コードなどの出力の改善要求	999
エラー・問題解決	エラー文や問題状況の共有と解決要求	551
知識の獲得・質問	知識向上のための質問	151
作業の代行	機械的作業の代行	149
出力・コードの説明要求	取得した情報の説明要求	319
理解・進捗の確認	ユーザの理解や現状の進捗の確認	71
その他	誤送信や謝辞などの例外的入力	89

分類の結果，表 2 に示すように，プロンプトが分類され，セッション内では平均して 5.1 回のプロンプト入力が行われたことがわかる．

4.2.2 入力プロンプトの目的別の分類

プロンプトデータの目的ごとの分類では，より詳細にユーザの目的意識を抽出し，適切なカテゴリを作成するためにデータに基づいた概念抽出を目的とした木下による修正版グラウンデッド・セオリー・アプローチ [27] と呼ばれる質的分析の手法を参考に分類項目の作成を行った．実際の分類項目の作成手順を以下に示す．

1. いくつかのデータを基に分類項目を作成する
2. 作成された分類ごとにワークシートを作成し，その定義や具体例を記入する
3. その他のデータに対しても，作成された分類項目に基づいて分類を行う
4. 分類中に例外が発生したら分類項目の再検討を行う
5. 分類項目が定着し，分類が終了するまで 2 から 4 を繰

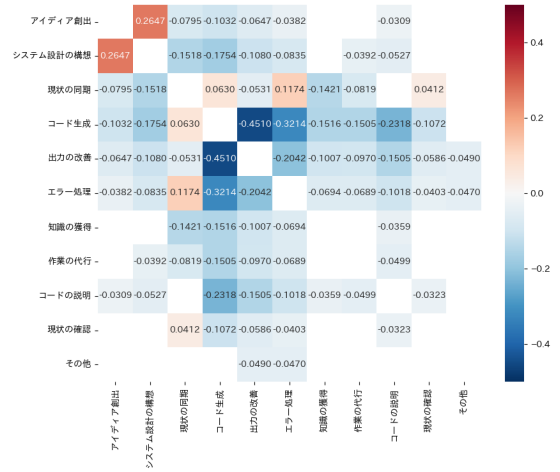

図 3: 同一プロンプト内の共起性．それぞれのラベルを並列に計算するため，対角線に対して対称な対称行列となっている．

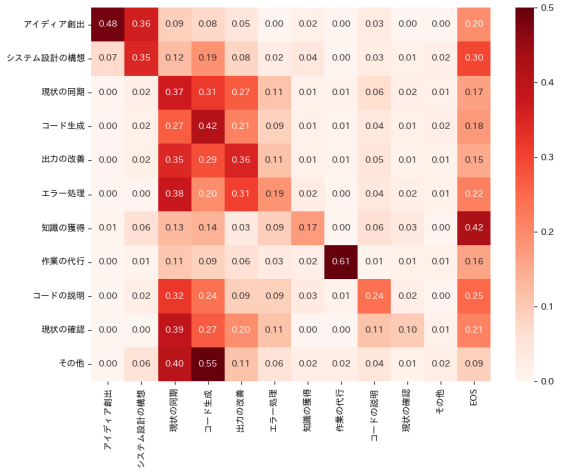

図 4: セッション内の前後のプロンプトの目的別ラベルに関する条件付き確率のヒートマップ．縦軸が前のプロンプトで横軸が後続のプロンプトに対応し，横軸内の EOS はセッションの終わり (End of Session) を表す．

り返す

以上の手法を用いて理論的飽和の状態[*4]となるまで分類作業と分類項目の検討を繰り返した．分類段階では，ユーザの複雑な目的意識を抽出するために，一つのプロンプトに対し，複数の分類 (マルチラベリング) を許容した．

分類の作業を進める中で複数の分類ラベルの該当箇所の重複や分類項目の定義との不一致があった場合には分類項目の再構成や再定義を行い，最終的に表 3 に示す 11 の分類項目が作成された．

これらの分類項目の作成においては客観性を担保するため，研究実施者二名での議論を行い，項目決定後の分類は筆頭著者が一人で行った．

5. データの分析

5.1 目的ごとの利用の調査

利用者のプロンプト入力の傾向を分析するため，プロンプトに与えられた目的別ラベル同士の関連性を調査した．

5.1.1 同一プロンプト内の目的別ラベルの関係性

まず，4.2 項で行った分類では同一のプロンプトに対して複数の項目への分類を許容するマルチラベリングを行ったため，同一のプロンプト内でどのような目的別分類が有意な関連性を持っているかを確認した．指標としてはカテゴリカルな変数間の相関関係を調べる際に用いられるカイ二乗検定 [12] を行った．また，カイ二乗検定によって有意に関連性が見られた ($p < 0.05$) 組に関しては，クラメールの連関係数 V を用いて，相関の大きさを算出した．

$$V = \sqrt{\frac{\chi^2}{n(k-1)}} \quad (\ n\text{: プロンプト数，}k\text{: 自由度}\ ) \quad (1)$$

[*4] 新たな分類や概念が生成されなくなり，分析上の問題点や例外処理がなくなった状態

図 3 では，相関の大きさとともにラベル同士に正の相関がある (有意に共起性が高い) か，負の相関がある (有意に共起性が低い) かを示すために，以下に示す相関比の符号を掛け合わせた数値を示している．

$$r = \frac{p_{xy} - p_x p_y}{\sqrt{p_x(1-p_x)p_y(1-p_y)}} \quad (2)$$

以上を元に，特に高い関連性が見られた組を表 4 にまとめる．

この結果から，利用者が一つのプロンプト内で同時に利用される目的の組み合わせと，異なるプロンプトによって達せられる目的の組み合わせが確認できる．特に，共起性の高い組を見ると，システム設計の構想において AI によるアイディア出しを行うことやエラー処理，コードの記述において AI に対してディレクトリ構造やコード状況を伝える現状の同期を多く行うことがわかる．

5.1.2 前後のプロンプトにおける目的別ラベルの関係性

また，5.1.1 項に引き続き，セッション内の前後のプロンプトの関連性も確かめるため，目的別ラベルの前後関係における共起性を以下に示す条件付き確率によって算出した (図 4)．条件付き確率による共起性の評価では，プロンプトの前後関係を考慮した評価が可能となる一方，「その他」ラベルのように出現確率 ($P(x)$) の小さいラベルに関連する確率が過大に評価される傾向を持つため，ラベル内での比較に留まるが，同一プロンプト内の調査と併せて「アイディア創出」と「システム設計の構想」に強い関連が見られることが分かる．また，表 3 にも示したように同一プロンプト内の調査では負の相関を示したラベル (コード生成，出力の改善，エラー処理) に関して前後関係においては一定の共起性が確認できた．これはこれらの目的のプロンプトが同時に利用されることが少ない一方で連続して用

表 4: 同一のプロンプト内で特に高い関連性が見られた目的別ラベルの組み合わせ．V はクラメールの連関係数．

有意に共起性が高い組	V	有意に共起性が低い組	V
アイディア創出，システム設計の構想	0.26	コード生成，出力の改善	0.45
現状の同期，エラー処理	0.11	コード生成，エラー処理	0.32
現状の同期，コード生成	0.06	コード生成，コード説明	0.23
現状の同期，現状の確認	0.04	出力の改善，エラー処理	0.20

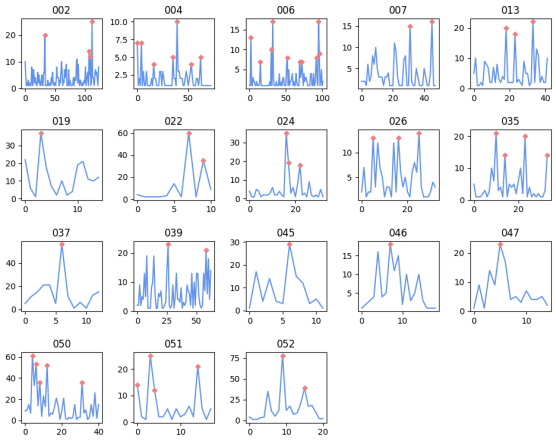

図 5: データ数が 100 件以上であった利用者の時系列順のセッション長．横軸はセッションのインデックスを表す．赤の菱形はセッションの長さが外れ値として検出されたセッション．

いられる傾向にあることを示している．これらに「現状の同期」を加えた 4 ラベルは表 3 からも分かるように出現数が多く，生成 AI を用いたコーディングにおいて中心的に利用される目的であり，プロンプト内・プロンプト間での利用数を調査することで，ラベル同士の強い関連性も確認できた．

5.2 セッションごとの利用の調査

セッションごとの調査では，利用の傾向を観察するため，セッション数が 100 件以上であった利用者のデータを用いて個人ごとに分析を行った．まず，生成 AI を用いたシステムの作成の過程においてセッションの長さがどのように変化するか調査を行った．すると，図 5 に赤の菱形で示したように長さが一時的に非常に大きくなるセッションが存在する傾向にあることが確認できた．このセッション長が増大する期間の生成 AI の利用について，その他のセッションとの利用方法の差異を調査し，その原因や特性を探る．増大したセッション長の抽出には，以下に示す各データの第三四分位数 (Q_3)，四分位範囲 (IQR) による外れ値判定を利用した．

$$outlier > Q_3 + 1.5 \times IQR \qquad (3)$$

以降，異常値判定により抽出されたセッショングループを

表 5: グループ I, II 間で利用の割合に有意な差 ($p < 0.05$) が見られた目的別ラベル．半数の利用者においてグループ I, II 間で利用に有意な差があった

目的別ラベル	割合が大きいグループ	利用者
出力の改善	グループ I	P2, P22, P24, P35, P50
コード生成	グループ I	P6, P35, P47
エラー処理	グループ I	P13, P37, P50
エラー処理	グループ II	P6
現状の同期	グループ I	P24

グループ I，その他のセッショングループをグループ II として表記する．セッショングループ I, II について利用方法に差異があるか調査するために表 3 に示した目的別ラベルの構成 (割合) を比較し，カイ二乗検定によって有意に差異が存在する ($p < 0.05$) ラベルを特定した．

表 5 に示すように 11 の目的別分類の中でも実際のコードの記述に大きく関係する 4 分類 (コード生成・現状の同期・エラー処理・出力の改善) においてセッションが長いグループ I とセッション長が短いグループ II 間で利用する頻度に有意な差があることがわかった．また，P6 の「エラー処理」を除いた他の目的別ラベルに関してはグループ I での利用が増大していた．P6 の「エラー処理」に関しては「コード生成」においてグループ I での利用が有意に大きくなっており，表 4 にあるように「コード生成」と「エラー処理」は共起性が低い組み合わせであるため，結果として「エラー処理」の割合が低くなっていることが考えられる．

このように，セッション長が増大するグループ I では有意な変化が見られた目的別ラベルから生成 AI によるコーディングが活発に行われていると捉えることができる．

6. 考察

6.1 対話型 AI を用いたコード記述におけるセッションの利用について

5.2 項ではセッション長の時系列データから，セッションの利用方法をセッション長が局所的に増大するセッショ

ングループ I とその他のグループ II に区分し，利用目的の比較を行った．結果としてコードの記述に関連するような「コード生成要求」や「エラー・問題解決」などの目的のプロンプトが多く入力されていることが分かった．

この結果に関して, Barke ら [2] が行った GitHub Copilot を用いたコーディングに対する質的調査の中で，生成 AI の利用について exploration mode (解決策が明らかになっていない状態で生成 AI を積極的に用いてコーディングをする時間) と acceleration mode (生成 AI を利用しながらも利用者自身が積極的にコーディングをする時間) の 2 つに分類できるとしており，今回観察されたセッション長が局所的に増大するグループ I ではコード記述に関連したプロンプトが有意に頻繁に利用されていることからも exploration mode に該当する利用段階であると考えられる．

グループ I の利用段階では表 5 で示した有意に利用が増えた目的ラベルから，「コードの生成」だけでなく「出力の改善」や「エラー処理」などの期待した結果が得られなかった際に用いられるプロンプトの利用が増加しており，セッション長が増大する原因としては，積極的なコード生成や，連続したエラー処理・コード修正などが考えられる．また，「コードの生成」,「出力の改善」,「エラー処理」の目的ラベルは表 4 に示したように，「現状の同期」と高い共起性を示しており，グループ I のような生成 AI 主導の実装段階においてファイル内容の提示や直面している問題・状況の共有など「現状の同期」が重要な役割を果たすと考察できる．この「現状の同期」に関して，講義の最終回後に行ったアンケート調査の中でも「ChatBot を利用する上で行った工夫点」に関して

> コードをベタッと貼り付けるなど、長い入力を与えるようになった。(P2)

> やりたいことやこれまでに書いたコード、エラー文など背景情報をなるべく多く入力した。回りくどいくらいに丁寧にやりたい処理を伝えた (P33)

のように「現状の同期」に言及している回答が多く得られた．

6.2 生成 AI を用いたシステム開発・コーディング支援について

これまで，様々な観点から生成 AI の利用の実態を調査・分析してきたが，それらの結果をもとにここでは生成 AI を用いたシステムの開発の支援について考察を行う．

6.2.1 現状の同期の重要性

プロンプト同士の関連性や，アンケートの結果から生成 AI に対する詳細な現状の同期や生じた問題の共有の重要性が明らかになった．特に，システム開発のような長期のコーディングを伴う場合には実装ファイルやディレクトリ構造が複雑かつ長大なものになりやすく，それらの正確な共有は難しくなる．実際，アンケートの中でも

> 長いコードを全行コピペすると期待通り応答しない時があったので、修正の必要のある箇所だけをコピペするなどした。(P9)

といった回答も得られており，現状の共有を容易に行う支援システムの設計が重要であると考える．その点では，本研究でも取り扱った ChatGPT などのような対話型 AI よりも GitHub Copilot などのようにエディタ上で動作し，インラインでコーディング支援が行える設計の方がより現状の共有という観点では有用であると言える．また，アンケートへの回答の中には，

> ChatGPT のコードで正しく動いた場合はあまりコードの意味を確認しないままになってしまうこともあった。(P33)

> 全体像を自分で把握していない。(P29)

といったように，ユーザの現状把握が追いつかない事例も報告された．目的別分類においても「現状の確認」や「コードの説明」の要求を目的としたプロンプトの利用が見られ，今後，生成 AI のコード生成精度の向上やコード作成支援技術の発展に伴い，利用者側がシステム実装を管理する立場として，その現状を把握することへの需要も高まるのではないかと考察する．実際，Jonsson ら [9] の AI を用いたプログラミング学習に関する実態調査の研究においても人間が実装状況を把握し，管理することが重要であると報告されている．このことを踏まえ，特にシステム開発のような大規模な制作過程においてはユーザと生成 AI それぞれの現状の同期機能を支援する仕組みが必要となると考える．

6.2.2 利用者のコーディング方針

講義中の生成 AI の利用による利用者自身の変化として，アンケート内では

> ChatGPT がエラーのない、期待する挙動を示す完璧なコードを書くまで対話を続けるのではなく、あくまで ChatGPT の書いたコードをベースにして自身でコードをリファクタしたり、あるいは必要な部分だけを抜き出して自身のコードに入れるようになった。(P23)

といった回答や

> コーディングがコーディングというよりもコードレビューに近くなったと思う。自分でどんどん書いていくというよりは、出力が正しいか、無駄がないかというのを気にする時間が多かったと思う。(P2)

といった認識の変化が報告された．このようにコードを「書く」ことよりも「確認，評価」するという認知的プロセ

スが重要になってきていることは関連研究の中でも言及されている [18]．このような変化に対し，AI の出力に対する評価や修正に焦点を当てた開発環境の可能性や支援方法を検討する必要がある．

7. まとめ

本研究では，情報可視化システムの制作を取り上げ，システムのプロトタイピングの初期段階から実装までの生成 AI の利用を観察し，プロンプト・データの分析を行った．本研究の対象は東京大学に在籍する学生であり，本研究で得られた結果の一般化には更なる調査が必要になると思われるが，目的別の分類では 11 種のラベルによってマルチラベリングを行い，4400 件のデータセットが得られた．こちらは今後の更なる研究や AI を用いたコーディング開発の支援システムに有効に利用されると考えられる．また，情報検索の研究分野で用いられるセッションの観点を取り入れ，新たな視点による分析と AI の利用に異なる段階が存在することを確認した．また，定量的な分析結果とインタビュー結果を照らし合わせ，システム開発における AI とユーザの状況の同期が不十分であることが確認され，支援の必要性が考察された．

謝辞

本研究の実施，本項の執筆に際し，アドバイスや相談に乗ってくださった研究室のメンバーの皆さん，実験に参加してくださった方々に深くお礼申し上げます．

参考文献

[1] Bansal, G., Wu, T., Zhou, J., Fok, R., Nushi, B., Kamar, E., Ribeiro, M. T. and Weld, D.: Does the whole exceed its parts? the effect of ai explanations on complementary team performance, *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pp. 1–16 (2021).

[2] Barke, S., James, M. B. and Polikarpova, N.: Grounded copilot: How programmers interact with code-generating models, *Proceedings of the ACM on Programming Languages*, Vol. 7, No. OOPSLA1, pp. 85–111 (2023).

[3] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A. et al.: Language models are few-shot learners, *Advances in neural information processing systems*, Vol. 33, pp. 1877–1901 (2020).

[4] Buçinca, Z., Malaya, M. B. and Gajos, K. Z.: To trust or to think: cognitive forcing functions can reduce overreliance on AI in AI-assisted decision-making, *Proceedings of the ACM on Human-Computer Interaction*, Vol. 5, No. CSCW1, pp. 1–21 (2021).

[5] Denny, P., Leinonen, J., Prather, J., Luxton-Reilly, A., Amarouche, T., Becker, B. A. and Reeves, B. N.: Promptly: Using Prompt Problems to Teach Learners How to Effectively Utilize AI Code Generators, *arXiv preprint arXiv:2307.16364* (2023).

[6] Fajkovic, E. and Rundberg, E.: The Impact of AI-generated Code on Web Development: A Comparative Study of ChatGPT and GitHub Copilot (2023).

[7] Jansen, B. J., Spink, A., Bateman, J. and Saracevic, T.: Real life information retrieval: A study of user queries on the web, *Acm sigir forum*, Vol. 32, No. 1, ACM New York, NY, USA, pp. 5–17 (1998).

[8] Jiang, E., Toh, E., Molina, A., Olson, K., Kayacik, C., Donsbach, A., Cai, C. J. and Terry, M.: Discovering the syntax and strategies of natural language programming with generative language models, *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pp. 1–19 (2022).

[9] Jonsson, M. and Tholander, J.: Cracking the code: Co-coding with AI in creative programming education, *Proceedings of the 14th Conference on Creativity and Cognition*, pp. 5–14 (2022).

[10] Kazemitabaar, M., Hou, X., Henley, A., Ericson, B. J., Weintrop, D. and Grossman, T.: How novices use LLM-based code generators to solve CS1 coding tasks in a self-paced learning environment, *arXiv preprint arXiv:2309.14049* (2023).

[11] Kong, A., Zhao, S., Chen, H., Li, Q., Qin, Y., Sun, R. and Zhou, X.: Better zero-shot reasoning with role-play prompting, *arXiv preprint arXiv:2308.07702* (2023).

[12] Li, C.-X.: Exploiting label correlations for multi-label classification (2011).

[13] Liu, Y., Deng, G., Xu, Z., Li, Y., Zheng, Y., Zhang, Y., Zhao, L., Zhang, T. and Liu, Y.: Jailbreaking chatgpt via prompt engineering: An empirical study, *arXiv preprint arXiv:2305.13860* (2023).

[14] Long, D. and Magerko, B.: What is AI literacy? Competencies and design considerations, *Proceedings of the 2020 CHI conference on human factors in computing systems*, pp. 1–16 (2020).

[15] McNutt, A. M., Wang, C., Deline, R. A. and Drucker, S. M.: On the design of ai-powered code assistants for notebooks, *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pp. 1–16 (2023).

[16] Myers, C., Furqan, A., Nebolsky, J., Caro, K. and Zhu, J.: Patterns for how users overcome obstacles in voice user interfaces, *Proceedings of the 2018 CHI conference on human factors in computing systems*, pp. 1–7 (2018).

[17] Raychev, V., Vechev, M. and Yahav, E.: Code completion with statistical language models, *Proceedings of the 35th ACM SIGPLAN conference on programming language design and implementation*, pp. 419–428 (2014).

[18] Sarkar, A., Gordon, A. D., Negreanu, C., Poelitz, C., Ragavan, S. S. and Zorn, B.: What is it like to program with artificial intelligence?, *arXiv preprint arXiv:2208.06213* (2022).

[19] Stol, K.-J., Ralph, P. and Fitzgerald, B.: Grounded theory in software engineering research: a critical review and guidelines, *Proceedings of the 38th International conference on software engineering*, pp. 120–131 (2016).

[20] Vaithilingam, P., Zhang, T. and Glassman, E. L.: Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models, *Chi conference on human factors in computing systems extended abstracts*, pp. 1–7 (2022).

[21] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. and Polosukhin, I.: Attention is all you need, *Advances in neural information processing systems*, Vol. 30 (2017).

[22] Wang, Y., Shen, S. and Lim, B. Y.: RePrompt: Automatic Prompt Editing to Refine AI-Generative Art

Towards Precise Expressions, *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pp. 1–29 (2023).

[23] White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J. and Schmidt, D. C.: A prompt pattern catalog to enhance prompt engineering with chatgpt, *arXiv preprint arXiv:2302.11382* (2023).

[24] White, J., Hays, S., Fu, Q., Spencer-Smith, J. and Schmidt, D. C.: Chatgpt prompt patterns for improving code quality, refactoring, requirements elicitation, and software design, *arXiv preprint arXiv:2303.07839* (2023).

[25] Xie, Y., Pan, Z., Ma, J., Jie, L. and Mei, Q.: A prompt log analysis of text-to-image generation systems, *Proceedings of the ACM Web Conference 2023* (2023).

[26] Yin, P. and Neubig, G.: A syntactic neural model for general-purpose code generation, *arXiv preprint arXiv:1704.01696* (2017).

[27] 木下康仁：修正版グラウンデッド・セオリー・アプローチ (M-GTA) の分析技法，富山大学看護学会誌， Vol. 6, No. 2, pp. 1–10 (2007).