

AIで学生は創造的になれる（プログラミング）

植田 康孝*・石川 妃葉**

要 旨

これまで人間がコンピューターに命令するにはプログラム言語が必要であったが、チャットGPTでは人間の自然な言語で指示を与えることが出来るようになった。これまでのコンピューターはプログラムを知っている人だけが効率良く利用できたが、それは限られた人たちのものでしかなかった。AIのプログラミングに使われることの多いPythonやシステム開発に使われるJava、数値解析に使われるMATLABなどの使用言語がある。しかし、大規模言語モデル（LLM）によりコンピューターが人間の言葉（自然言語）により、C言語も、パスカルも、Javaも、Pythonを動かすことが出来るようになった。テクノロジー習熟の文理格差を生まないためにも、情報技術教育を充実させる必要がある。一部のエンジニアや研究者、理系学生のみがテクノロジーを扱えるのではなく、文系学生でも使えるようにして行くことが、文系教育を担う教員に課せられた使命である。

対話系AIに「ブラウザで動くブロック崩しのプログラミングをHTML、CSS、Javaスクリプトで書いてください」と頼むとコーディングしてくれる。途中で止まっても、「続けて」と書くと、書いてくれる。バグがあってもマニュアルで修正すれば良く大幅に時間が短縮されるようになった。歴史上初めて起こる「コンピューターの民主化」「特殊技能の大衆化」である。今までテキスト・マニュアル通りに書かなかった場合、例えば先生の言う通りに入力しても「カンマ（,）」と「ピリオド（.）」打ち間違えるだけでコンピューターは動かなかった。そのため、学生はエラーが怖くプログラミングにチャレンジ出来なかった。コンピューター言語の概念や用語は忘れ易いが、自然言語は忘れ難いため、普遍的である。本稿では「大規模言語モデルや生成AIを使えば、C言語やパスカル、JavaやPythonなど既存のプログラミング言語を習得するよりも簡単にコーディングできる」という仮説を立て検証した。プログラマーであり人工知能研究者である清水（2023）は、「日本企業がAI領域に参入して行くためには、ChatGPTを使ったプログラミング学習を積極的に進めた方が良い。ハードルの高い既存のプログラミング言語を使って習得するよりもはるかに簡単である」と主張する⁽¹⁾。プログラマーの間でも「時間を節約できる」と実際に使われ始めている⁽²⁾。これまでプログラミングやデータサイエンス教育を推進して来た教員からは、誤情報の発生や、教師データのバイアス問題、セキュリティなど生成AIの危険性が指摘される。しかし、過去の歴史が示しているように、この大きな技術の流れにあらがうことは難しい。その欠陥や脅威におびえて多くのネガティブな教員を真似て傍観者で終わるのか、そのポジティブな可能性を信じて生成体験する当事者になるのか。変化の速い時代においては、学生は多くの技術に触れ、自ら試行錯誤すること（AIによる様々な生成体験）で、技術の健全な発展に寄与してくれる人材になってくれると期待したい。2023年8月15日、岸田首相がプログラミング学習している映像が放送されたが、シンガポールや台湾のトップがプログラマーであることを捉えれば、プログラミング体験している方が望ましい。

キーワード：コンピューターの民主化 特殊技能の大衆化 大規模言語モデル（LLM） GitHub Copilot テクノブ
ォビア ノーコード AI 製作 学習 AI AI2.0 文理融合 総合知

2023年11月30日受付

* 江戸川大学 マス・コミュニケーション学科教授
理学博士（国際情報通信学）

** 江戸川大学 植田ゼミ第15期 令和5年度卒業生
総代

1. コード生成AIをめぐる動向

1.1. 生成AIのコーディング利用

コーディングとは、プログラミングの作業の一

部を指し、「プログラミング言語を用いてプログラミングコードを書いていく作業のこと」であり、「コードを書くこと」のみを示す⁽³⁾。そのため、プログラミングの作業にある「設計」「テスト」「バグの修正」「納品」等の作業は含まれない。プログラミング技術習得のためには、「アウトプット」が重要とされる。コーディングの経験を積むことにより、実践的な能力を身につけることができると判断されたため、2020年2月、文部科学省による「小学校プログラミング教育の手引」では、「企業と連携しながら行う授業実践を踏まえた指導例」を追加する改訂がなされ、プログラミング教材を活用した指導を行う旨の内容が記された⁽⁴⁾。

コード生成 AIは、自然言語処理や機械学習を用いてプログラムコードを生成してくれるAIを指す。主にゼロからコードを生成するものと、既存のコードの修正を行ってくれるものに分かれ、ソフトウェア開発を効率化し、エラーを減らすことが期待される。コード生成はソフト開発者支援でもAIの活用が進む。フランス出身のクレマン・デラングCEOが設立したハギングフェイスはソフト化共有サイト「ギットハブ（GitHub）」のAI版として、エンジニアのコミュニティの基盤となりつつある。GitHub Copilotは、GitHubが提供しているサービスで、プログラマーが記載したコメントや関数から、AIが最適なコードを推測し提案してくれる。GitHubは2023年1月にユーザーが1億人に到達したことを発表、大半がエンジニアであることを考えると、サービス内にコード生成AIが導入されたことは大きなインパクトがあると考えられる（図1）。また、メタ（Meta）の「Code Llama」は、Llamaをベースに、コードに特化したデータセットで訓練したモデルで、最高水準の評価「SOTA（state-of-the-art）」を達成している。

1.2. パナソニック・コネクットの事例

パナソニック・コネク트가、ソフト開発/コーディングに生成AIを活用する検証を行った。2023年7月10日から「GitHub Copilo」を導入し

生成系AIの発展 達成年は更に早まる見込み

	2020	2022	2023	2025	2030
テキスト	コピーライティング アイデア出し	長文生成 文章の編集	各業界の用途に最適化 した文章作成	平均的な人間の文章能力を上回る	プロのライター の文章能力を上回る
コード	複数行のコード生成	大規模なコード生成 高精度なコード編集	専門的な開発要件 に対応	テキストからプロダクト を生成する	フルスタックエンジニア の能力を上回る
画像		アート ロゴ 写真	プロダクトのモック アップの生成	プロダクトデザインの完成稿 を作成	プロのデザイナーの 能力を上回る
動画		簡単な動画素材の生成	汎用的な動画素材の作成	動画の生成	ゲームの生成 映画の生成

図1 生成系AIの発展

出所：株式会社デジタルレシビ（2023）⁽⁵⁾

活用に取り組んだ。エディタ上で自然言語コメントを書くだけで、ソースコードを生成する。対話型かつリアルタイムに開発者の意図を汲んでコードを提案してくれる。開発言語はPythonが1位で、JavaScript、C#、C++が多かった。

1.3. LINE ヤフーの事例

LINE ヤフーは2023年10月中旬からソフトウェア開発に生成AI「GitHub Copilot」を全面的に導入した。社内のエンジニア約7,000人全員が開発したい機能や動作を入力すると必要なコードを作成する。コード生成AIを導入した結果、コード作成に費やす作業時間を平均で1日当たり約2時間減らすことが出来た。開発中心のエンジニアでは3～4時間短縮できた。LINE ヤフーが生成AI導入で減らせる作業時間は2割強に相当した。

1.4. NTT データの事例

NTT データグループは生成AIをシステム構築へ適用した。作業工数は人手に比べて7割削減できたが、秘訣はプロンプト（指示文）を上手く使うことにあった。変換前のソースコードに基づき生成AIは新たな開発言語によるソースコードを提案する。担当者は変換後にテストを行い、思い通りに変換されていないものについて人手で修正した。変換精度が向上し、7割の工数削減につながった。

1.5. 米国ハーバード大学のコード生成 AI

ハーバード大学は2023年秋から「ChatGPT」搭載の人工知能（AI）を使用して、コンピューターサイエンスの入門コース（Computer Science 50：CS50）の指導を支援すると同時に、授業助手（TA）の手間を解消した。コード作成課題の内容や、コードスタイルの改善方法を説明したり、良くある質問に回答したりすることで、学生に対してパーソナライズされた支援を提供できるようになった。TAや教授に助けを求められない場合、学生が行き詰まったりやる気をなくしたりすることを防げる⁽⁶⁾。

2. AIによるコーディング検証

専門ゼミナールでは、AIとプログラミングの可能性に着目し、ChatGPTを用いて、プログラミング教材の問題を活用し、その問題のコーディングを実践した。模範解答との比較を行うことにより、AIによるコーディングの可能性を検証した。

2.1. プログラミング教材

「プログラミング教材」とは、プログラミングの学習を支援するために作られた教育資料や学習コンテンツを示す。初心者から上級者まで、レベルに沿った教材が提供されている。オンライン形式、テキスト形式、チュートリアルやドキュメンテーション形式、オンラインプラクティスプラットフォーム形式など学習者が実践的な経験を積むこと出来るプログラミング教材が存在する。プログラミング教材の中からテキスト形式を選択し、各プログラミング言語の入門参考書に掲載された問題を活用し、筆者（石川）がChatGPTを用いてコーディングを行い比較した。扱った問題集は、表1の通りである。

2.2. コーディングの実践と模範解答との比較

「JavaScript」、「Python」、「Java」、3つのプログラミング言語を用い、ChatGPTによるコーディングを実習した。各プログラミング言語の入門

表1 プログラミング教材 各プログラミング言語参考書

プログラミング言語	参考書
JavaScript	田中賢一郎 (2019) 『ゲームで学ぶ「JavaScript入門」』 インプレス
Python	Srini Devadas 黒川利明 訳 (2018) 『問題解決のPythonプログラミング—数学パズルで鍛えるアルゴリズム的思考』 オライリー・ジャパン
Java	新谷虎松 (2014) 『改訂 Java による知能プログラミング入門』 コロナ社

出所：筆者（石川）が独自に作成。

参考書に掲載された模範解答を用い、ChatGPTを用いコーディングした。

(1) 表1に示した各参考書に記載のあるプログラミング問題を用い、PC版ChatGPT「<https://chat.openai.com/auth/login>」によるコーディングを行った。

(2) ChatGPTに対するテキストは「『JavaScript/Python/javaで〇〇をコーディングしてください。』」と、プログラミング問題の題材（〇〇とする部分）以外は、共通したテキストで対話することで統一した。

(3) コーディング実践中は、参考書に掲載された模範解答や他コーディング事例が掲載されたサイトを閲覧しない。一方、コーディング比較中は、参考書に掲載されている模範解答のみを閲覧し、それ以外の事例を閲覧せず比較した。

(4) 技術者によるコーディングの癖に関する言及は省くようにした。

(5) コーディング実践は、問題に対するコーディングを求め、図1、図3、図5をエビデンスとして記録した。コーディング比較は、ChatGPTが生成したコーディングのみを模範解答と比較し、双方で一致している点を緑のラインで示し、図2、図4、図6にエビデンスとして記録した。

2.2.1. JavaScript

JavaScriptでは、「Dateオブジェクトのコーデ

イング」を実践する。問題内容を要約すると、「JavaScriptがあらかじめ持つオブジェクト（組み込みオブジェクト）を持っている。その中でもDateオブジェクトは日付や時刻を扱うためのオブジェクトであり、日付や時刻の取得や設定には数多くのメソッドが必要になる⁽⁷⁾。そのメソッドを理解するためにDateオブジェクトのコーディングを行う」という問題である⁽⁸⁾。Web上でも日付や時刻を取得できるようにするプログラムのコーディングを実践する。

ChatGPTに対し、筆者が「JavaScriptで日時・時刻の取得ができるようコーディングしてください」と入力し対話を行った結果、ChatGPTは「以下はJavaScriptでDateオブジェクトを使用して日付や時刻を操作する例です」という対話を生成した後、JavaScriptを用いたDateオブジェクトのコーディングが表示された。コーディング生成までには1分を要した。ChatGPTが生成したコーディング結果は、図1に記録した通りである。参考書に記載のある模範解答との比較を行ったところ、図2に示した通り一部に一致点（緑ライン）が見られただけで、ChatGPTが生成した

コードはすべて「オリジナル」と言える。

比較すると、模範解答の場合、HTML文書を用いてコーディングするが、ChatGPTの場合、HTML文書の構築を全て省略してコーディングを始め、メソッドが異なっていた。コーディングにおいて、HTML文書は、ブラウザに対して構文を伝えるために使用され、最初の行に記述される文書型宣言のため必要なコードである。この相違を発見した時点でChatGPTが生成したコードだけでは、正常に動かないことを予測できた。

しかし、ChatGPTが生成したコードの方が、文系学生にとって理解し易い。参考書の場合、英語の文字列がひたすら綴られているだけであり、何を示しているかを理解できない。ChatGPTの場合、各コードがブロックとして区切られており、灰色の文字で「コードが何を示しているのか」が示されており、一目でコードの内容を理解できる。また、模範解答よりもChatGPTが生成しているコードが中学生英語レベル程度で平易であるため、簡単に理解できる。

更に、ChatGPTは、模範解答には記載がなかったコードまでも生成してくれた。「各要素を取

```

// 以下は、JavaScriptでDateオブジェクトを使用して日付や時刻を操作する例です。

// Dateオブジェクトの生成
const date = new Date();

// 特定の日時を指定してDateオブジェクトを生成
const specificDate = new Date(2023, 4, 31, 15, 30, 0); // 2023年5月31日 15:30

// Dateオブジェクトから各要素を取得
const year = date.getFullYear();
const month = date.getMonth() + 1; // 0から始まるため、+1する
const day = date.getDate();
const hours = date.getHours();
const minutes = date.getMinutes();
const seconds = date.getSeconds();
const milliseconds = date.getMilliseconds();

// 特定の要素を設定してDateオブジェクトを変更
date.setFullYear(2024);
date.setMonth(8); // 0から始まるため、9月に設定
date.setDate(15);
date.setHours(12);
date.setMinutes(0);
date.setSeconds(0);

// Dateオブジェクトを文字列に変換
const dateString = date.toDateString(); // "Fri Sep 15 2024"
const timeString = date.toTimeString(); // "12:00:00 GMT+0000 (Coordinated Universal Time)"
const dateTimeString = date.toISOString(); // ローカルの日付と時刻に基づく文字列

console.log(year, month, day, hours, minutes, seconds, milliseconds);
console.log(dateString, timeString, dateTimeString);

```

図1 「Date オブジェクト コーディング結果」

```

// 以下は、JavaScriptでDateオブジェクトを使用して日付や時刻を操作する例です。

// Dateオブジェクトの生成
const date = new Date();

// 特定の日時を指定してDateオブジェクトを生成
const specificDate = new Date(2023, 4, 31, 15, 30, 0); // 2023年5月31日 15:30

// Dateオブジェクトから各要素を取得
const year = date.getFullYear();
const month = date.getMonth() + 1; // 0から始まるため、+1する
const day = date.getDate();
const hours = date.getHours();
const minutes = date.getMinutes();
const seconds = date.getSeconds();
const milliseconds = date.getMilliseconds();

// 特定の要素を設定してDateオブジェクトを変更
date.setFullYear(2024);
date.setMonth(8); // 0から始まるため、9月に設定
date.setDate(15);
date.setHours(12);
date.setMinutes(0);
date.setSeconds(0);

// Dateオブジェクトを文字列に変換
const dateString = date.toDateString(); // "Fri Sep 15 2024"
const timeString = date.toTimeString(); // "12:00:00 GMT+0000 (Coordinated Universal Time)"
const dateTimeString = date.toISOString(); // ローカルの日付と時刻に基づく文字列

console.log(year, month, day, hours, minutes, seconds, milliseconds);
console.log(dateString, timeString, dateTimeString);

```

図2 「Date オブジェクト 模範解答との一致点」

得」のコード以外、「特定の日時を指定してDateオブジェクト生成」「Dateオブジェクトを文字列に変換」「特定の要素を設定してDateオブジェクトを変更」など各ブロックのコードは模範解答ではなく、自動的にChatGPTが生成した。双方のコード数を確認すると、ChatGPTは20コードで、模範解答は30コードでChatGPTよりも1.5倍コードが多かった。JavaScriptにおいて、ChatGPTはHTML文書の構築を省く分、コーディングの文法よりもコードがもたらす機能性と分かり易さに重点を置いていた。

2.2.2. Python

Pythonでは、「ハノイの塔（『バラモンの塔』とも呼ばれる）のコーディング」を実践した。内容を要約すると、「ハノイの塔は数学パズルであり、規則に従って円盤を動かしているゲームである。ここでは再帰的に1つ減らして探索するというコーディングを理解するために、円盤の数をn枚として、円盤を動かす時に追加条件のあるプログラムのコーディングを行う」という問題である⁽⁹⁾。言い換えれば、数学パズルのハノイの塔を効率的に解くことが出来るプログラムのコーディングを実習した。

ChatGPTに対し、筆者が「Pythonで、ハノイの塔をコーディングしてください。」とプロンプトを入力し生成を指示した結果、ChatGPTは、筆者の意図とは別に自動的に生成するコード部分を絞り、「以下は、Pythonでハノイの塔を再帰的に解くためのコードです」という対話の後、Pythonを用いた「ハノイの塔の再帰解」のコードを表示した。生成コード数が少なかったため、僅か30秒で生成された。ChatGPTが生成したコーディング結果は図3に記録した通りである。

参考書に掲載された模範解答との比較を行ったところ、図4に示した通り、一致点（緑ライン）が見られただけで、ChatGPTが生成したコードは全てオリジナルであった。文法やコーディングの順番など類似する点が幾つか見られたが、コード内で示す事柄に対するメソッドが異なった。ChatGPTの場合、円盤の数を「n」とするのに対

以下は、Pythonでハノイの塔を再帰的に解くためのコードです。

```
python
def hanoi(n, source, target, auxiliary):
    if n > 0:
        # n-1枚の円盤をsourceからauxiliaryへ移動
        hanoi(n-1, source, auxiliary, target)

        # 最大の円盤をsourceからtargetへ移動
        print(f"円盤 {n} を {source} から {target} へ移動")

        # n-1枚の円盤をauxiliaryからtargetへ移動
        hanoi(n-1, auxiliary, target, source)

    # ハノイの塔の解法を呼び出す
    hanoi(3, 'A', 'C', 'B')
```

図3 「ハノイの塔 コーディング結果」

以下は、Pythonでハノイの塔を再帰的に解くためのコードです。

```
python
def hanoi(n, source, target, auxiliary):
    if n > 0:
        # n-1枚の円盤をsourceからauxiliaryへ移動
        hanoi(n-1, source, auxiliary, target)

        # 最大の円盤をsourceからtargetへ移動
        print(f"円盤 {n} を {source} から {target} へ移動")

        # n-1枚の円盤をauxiliaryからtargetへ移動
        hanoi(n-1, auxiliary, target, source)

    # ハノイの塔の解法を呼び出す
    hanoi(3, 'A', 'C', 'B')
```

図4 「ハノイの塔 模範解答との一致点」

し、模範解答の場合は「numRings」と示した。

ChatGPTが生成したコードは、ハノイの塔を解くために必要な手順を出力するコードであったが、模範解答にある出力までコーディング出来ないため、このコードだけではハノイの塔を解くことは不可能である。また、ハノイの塔における、再帰呼び出しで「棒に番号を付ける」ことが重要とされるが、ChatGPTの場合、番号を付けている形跡はないものの、最後のコードでハノイの塔の解法を呼び出すコードを生成することが出来ていた。双方のコード数を確認すると、ChatGPTの場合6コードで、模範解答の場合12コードでChatGPTより2倍コードが多かった。JavaScriptにおいて、ChatGPTは自動生成するコード部分を絞ってコーディングを行った分、コーディングのシンプルさに重点を置いていた。

2.2.3. Java

Javaでは、「簡単な電卓プログラムのコーディ

ング」を実践した。要約すると、「Javaの基礎を応用しやすいよう電卓プログラムのコーディングを行う」という問題である⁽¹⁰⁾。Javaの基礎を理解した上で応用問題の一例であるWeb電卓を作るため、簡単な電卓プログラムでコーディングを実践する。

ChatGPTに対して、筆者が「Javaで電卓プログラムをコーディングしてください」と入力し対話を行った。結果、ChatGPTは、「以下は、Javaで簡単な電卓プログラムをコーディングした例です」という対話を返した後、Javaを用いた簡単な電卓プログラムのコーディングを表示した。コーディング生成に1分を要した。ChatGPTが生成したコーディングは図5に記録した通りである。参考書に記載のある模範解答との比較を行ったところ、図6に示した通り冒頭の一部に一致点

（緑ライン）が見られただけで、ChatGPTが生成したコードは全てオリジナルであった。

使用したメソッドが異なっており、計算するコマンドが異なったコーディングであったため、模範解答と合致点は少ない。ChatGPTの場合、自動的にコードの冒頭に「import java.util.Scanner;」として、Javaのプログラムで「キーボードから情報を受け取る」ためのインポート文を生成しているが、模範解答にはインポート文の記載はなかった。また、ChatGPTが生成したコードは、文系学生にとって理解し易い。参考書の場合、英語の文字列がひたすら綴られているだけであり、それらが何を示しているか理解できない。一方、ChatGPTの場合、各ブロックのコードを端的に短いコードで示しているため、一目でコードの内容が何を示しているか予想できる。双方の

```

import java.util.Scanner;

public class Calculator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("最初の数値を入力してください: ");
        double num1 = scanner.nextDouble();

        System.out.print("演算子を入力してください (+, -, *, /): ");
        char operator = scanner.next().charAt(0);

        System.out.print("次の数値を入力してください: ");
        double num2 = scanner.nextDouble();

        double result = 0;

        switch (operator) {
            case '+':
                result = num1 + num2;
                break;
            case '-':
                result = num1 - num2;
                break;
            case '*':
                result = num1 * num2;
                break;
            case '/':
                if (num2 != 0) {
                    result = num1 / num2;
                } else {
                    System.out.println("エラー: 0で割ることはできません。");
                    return;
                }
            default:
                System.out.println("エラー: 無効な演算子です。");
                return;
        }

        System.out.println("結果: " + result);
    }
}

```

図5 「簡単な電卓プログラム コーディング結果」

```

import java.util.Scanner;

public class Calculator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("最初の数値を入力してください: ");
        double num1 = scanner.nextDouble();

        System.out.print("演算子を入力してください (+, -, *, /): ");
        char operator = scanner.next().charAt(0);

        System.out.print("次の数値を入力してください: ");
        double num2 = scanner.nextDouble();

        double result = 0;

        switch (operator) {
            case '+':
                result = num1 + num2;
                break;
            case '-':
                result = num1 - num2;
                break;
            case '*':
                result = num1 * num2;
                break;
            case '/':
                if (num2 != 0) {
                    result = num1 / num2;
                } else {
                    System.out.println("エラー: 0で割ることはできません。");
                    return;
                }
            default:
                System.out.println("エラー: 無効な演算子です。");
                return;
        }

        System.out.println("結果: " + result);
    }
}

```

図6 「簡単な電卓プログラム 模範解答との一致点」

コード数を確認すると、ChatGPTは36コードで、模範解答は80コードであり、ChatGPTの約2.2倍コードが多かった。

Javaにおいて、ChatGPTは、模範解答に掲載されたコードを大きく省略し、計算に必要なコードのみを生成した一方、指定がなかったクラスの呼び出しや電卓の表示など詳細なプログラムの部分に関するコードの生成を省いてコーディングを行った。Javaにおいて、ChatGPTは電卓の位置関係に対するコードを省いた分、電卓の根本的役割の計算が可能になれば問題ない機能に重点を置き、コード生成していた。

2.3. 検証結果

「ChatGPTを用いれば、プログラミング言語を全く知らない文系学生でも、自然言語処理（NPL）の力を活用してコーディング出来る」という仮説を立て、ChatGPTを用いてプログラミング問題を実習した。模範解答とコーディングの比較を行うことにより、AIによるコーディングの可能性を検証した。文系学生が入門書や参考書を習い経験しても、正しいコーディングよりも誤ったコーディングを行う可能性が高いことを前提とした。コーディングはプログラミングソフトウェア上の作業で行うが、作業自体は人間の手作業によって行われるため、誤ったコーディングを行わないとは断言できない。

人間だけでコーディングするよりもChatGPTによりコーディングを生成してもらった方が、はるかに短い時間で生成できる上、誤ったコーディングを行う可能性は低い。しかし、文系学生の場合、ChatGPTによりコーディングを生成したとしても、コーディングが正しいと判断する基準を持ち合わせないため、コーディング経験がなくともコーディングすることは可能であるが、コード確認に対する判定は人間自身が行う必要があるため、コーディングが出来、プログラムが正常に動くとは明言できない。現状では、プログラミングにおいて人間の作業を完全に代替できる技術はAIは実装していないため、学生はある程度プログラミング言語を学習し理解する必要がある。

表2 ChatGPTによるコーディング結果

プログラム言語	ChatGPTでの生成	模範解答との一致率 (%) * (一致数 ÷ 模範解答のコード数 × 100)
JavaScript	対応している○	26.6%
Python	対応している○	25%
Java	対応している○	2.5%

出所：筆者（石川）が独自に作成。

分析と検証の結果、「ChatGPTを用いれば、プログラミング言語を全く知らない文系学生でも、自然言語処理（NPL）の力を活用してコーディングできる。」という結果が導出された。しかし、ChatGPTが生成したオリジナルのコードだけで、完成形のコーディングを最初から最後まで行うことは、現状では難しい。間違ったコーディングか、正しいコーディングか、判断するのは人間の役割（AIリテラシーやプログラミング経験に基づく判断）として残る。

古川・酒井（2023）も同様に「生成された文章のファクトチェックは不可欠」と主張し、「既存のWeb上のデータを使っている以上、既存のコンテンツと似通ったものが生成されてしまう可能性もゼロではありません。（中略）必ず内容が正しいかどうかのチェックや、他人の権利を侵害していないかどうかの確認が必要になります。」とChatGPTの扱いを説明している⁽¹¹⁾。現状のAI技術の利用において、ChatGPTを活用すれば、文系学生がプログラミング知識が持たなくともコーディング出来るようになるが、ChatGPTが生成したコーディングでプログラミングが成功する（コード確認を行った上での完成形になる）と明言は出来ない。ChatGPTの活用には、「AI任せにしない」人間との「共存」の形であり、利用する側は「生成したい内容を明確にして生成されたものに対してファクトチェックを行う役割」、利用される側は「文章を生成する役割」、と双方の役割を分担し共創する必要がある。

3. 大規模言語モデル (LLM) 活用

次のプログラミング問題を大規模言語モデル (LLM) の AI に解かせてみると、どうなるか？

【プロンプト】

「あなたは水温を測る温度計のプログラムを作成しています。0 度以下になり水が凍る場合に警告を出すことにしました。水温 n 度が与えられるので 0 位かの場合「freeze」、1 以上の場合は「n」を出力してください」

【試行感想】

水温を測定して、指定された条件に従って警告や温度を出力するプログラム (Python を用いたコード) を 1 回のプロンプト指示で生成した。生成されたコードが機能するか検証するため、Google の「Colaboratory」を活用した。コードを実行した結果、図 7 のように「水温を入力してく

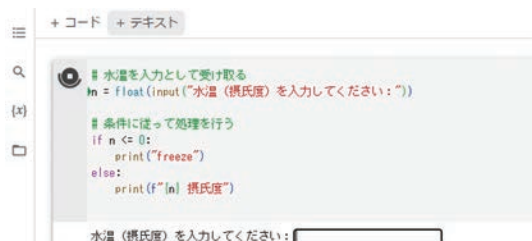


図 7 入力項目表示

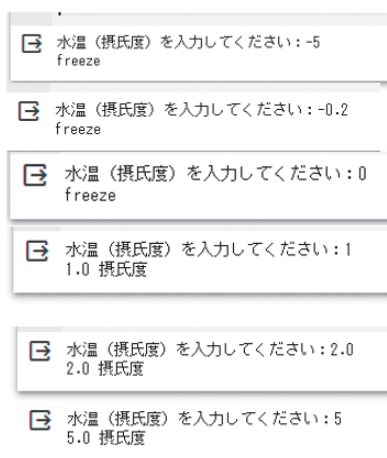


図 8 出力検証結果

ださい。」と入力項目が表示された。実際、正しく出力されるか確かめるため、「-5, -0.2, 0, 1, 2.0, 5」と 6 回入力を試した。実践した結果、図 8 の通り、「0 以下の場合 freeze, 1 以上の場合は n」と出しく出力していることが確認できた。よって、生成されたコードはプロンプトの要望通り正しく機能したと言える。また、コード生成に 1 分かかったが、入力出力に掛かる時間は 2 秒だった。つまり、このプログラミング問題に関しては、人間が 1 から考えてコードを入力するよりも AI の力を借りてコードを構築していく方がはるかに効率的であると言える。但し、この問題の模範解答が分からないため、AI が提示してきたコードだけが正しいとは言い切れない。そのため、AI が生成したコードを模範 (ヒント) にしつつ、人間が手を加え、AI を使用してコードの見直しや質向上に活用した方が役に立つと言える。

4. 生成 AI をプログラミング教育支援に活用

4.1. プログラミング支援 AI

「教えて MENTA くん」

ChatGPT の技術を用いて、スキルアップを支援する AI ツールである。LINE アカウントを友達に追加することで会話を開始でき、プログラミングに関する質問への回答や壁打ちが出来る。プログラミングの実装方法、利用ツールの提案を受ける、メンターを探す、技術に関するクイズを楽しむなど多彩な使い方が出来る⁽¹²⁾。

<https://line-up.app/oshiete-menta-kun/>

【筆者 (石川) の感想】

(試した質問)

- 今日のニュースは？ (テンプレートにあった質問)
- 動画生成 AI でオススメのツールは？ (自分で考えた質問)

(感想)

- 日頃から LINE を使用しているため、特に準備は不要。ワンクリックで「友だち追加」すれば利用でき簡単だった。不要になれば「ブロックできる」という単純さやポピュラーな

SNSで利用できる利点がユーザーへの負担を小さくした。

- 「今日のニュースは」という質問に対して、AI関連のニュースに絞って提示され1分程度で情報を手早く知れた。就活面接の準備として役立つだけでなく、就活後もAIに関する話題を簡単に得られるため、時間を節約して情報を得られる点で非常に便利で、今後も利用を続けたいと感じた。
- オススメツールに関しては、無料か有料かを配慮しオススメ生成ツールを提示してくれた。これを参考に卒業論文で活用すれば、他学生と同じツールを使わずに済み、執筆内容だけでなく用いるツールでも、他学生と差別化できると考えた。
- ChatGPTと明らかに異なる点は、ユーザーへの注意喚起を行うことである。ChatGPTに同じ質問をしたが、要件に合った提案を箇条書きで提示するだけだった。一方、「教えてMENTAくん」は、要件に沿った提示だけでなく「機能制限」「商用利用」「著作権」に関して、ユーザーに注意することを付け加える特徴が見られた。
- MENTAくんではプログラミングの壁打ちを試した。学習ロードマップの相談と、初心者でも直ぐに書けるコードの提案を受けた。初歩的な事を質問しても嫌な顔をされないのが、人間と違って良い部分とも言える。ただ、1日5回までの回数制限があるため、テンポよく壁打ちできる相手とは言えなかった。

5. ノーコードでAI製作

5.1. グーグル「Teachable Machine」

「Google Teachable Machine」とは、Googleが提供しているAI機械学習ツールで、ブラウザからサイトへアクセスするだけで使用できる。プ

ログラミングしなくても、画像や音声、身体の姿勢（ポーズ）などを学習させてそれらを判別するモデルを作成できる。モデル生成には、深層学習（Deep Learning）技術が使われており、複数の画像や音声から特徴点（似た部分）を抽出し学習を行う（図9）。

<https://teachablemachine.withgoogle.com/>

開いたら青色のボタン「使ってみる」をクリックする。

<https://teachablemachine.withgoogle.com/train/image>

実習では、学生全員に「グー」「チョキ」「パー」の画像をAIに認識してもらい学習データを作成して判別AIをプログラミングしてもらった。他には、「マスクある」「マスクなし」判定、オンライン試験時のカンニング判定、友達と協力して「Aちゃん」「Bちゃん」の顔認証、お茶を使って「A社」「B社」のブランド判定など自由に考えオリジナルAIを製作してもらった。

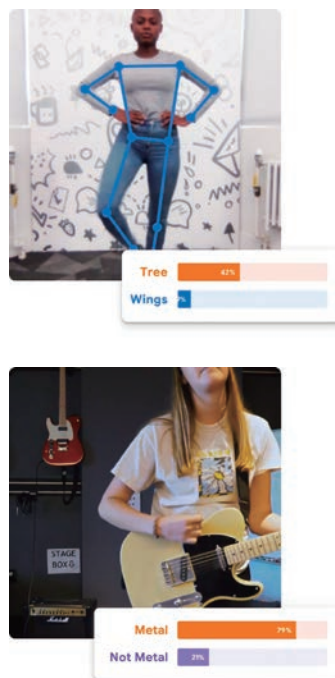


図9 グーグル「Teachable Machine」

【学生の感想①】

ウェブカメラで色々な角度から「グー」「チョ

キ」「パー」の形をした手を撮影してトレーニングするとAIに反映され、自分の手とじゃんけん出来ました。画面越しに自分の手とじゃんけんするのは不思議で面白かったです。AIとは、こうやれば作れるということを知ることが出来、追加で出された課題のAIも作りたくなりました（図10）。



図10 グーグル「Teachable Machine」(学生①)

【学生の感想②】

ソフトを使ってAIにグーチョキパーを学習させた。AIを作るのはもっと高度な技術が要と思っていたが、データを集めモデルを作成し評価・検証をひたすら繰り返す技術は根気や体力が要る作業だということが分かった。凄く簡単にAIを作れたので楽しく学べた。色々な人の画像を解析して楽しみたいと思った（図11）。



図11 グーグル「Teachable Machine」(学生②)

【学生の感想③】

手を微妙な形にしたらどうなるかというAIを作ったところ、全部にパーセントが出た。どの程度からグーチョキパーを識別して、手の形か指の本数か、そもそも指を認識しているかなど、気になることがたくさんあった。1番認識し易いのはグーだった。サンプル数を増やすことで精度が高まることが分かったので、様々なことに活用できると思います（図12）。



図12 グーグル「Teachable Machine」(学生③)

【学生の感想④】

AIを作ることは、難しい数式を組まないと出来ないといけないと思い、自分ではとても無理だと思っていました。しかし、画像データを撮影しながら楽しくAIを作れました。AIは凄いと思いました。貴重な体験をありがとうございました（図13）。

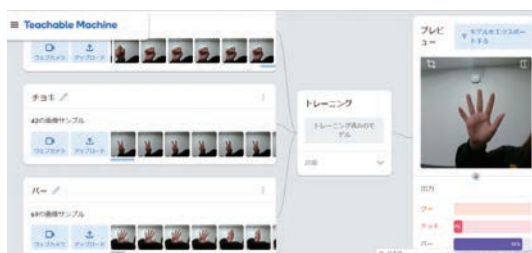


図13 グーグル「Teachable Machine」(学生④)

【学生の感想⑤】

指の本数の違いでやりましたが、きちんと正しく判別されました。パーセンテージも高く、正しい判別が出来ているので、凄いと思いました。面白かったのでメイクでも判別出来るか試しました。やはりAIは凄いなと感じました（図14）。



図14 グーグル「Teachable Machine」(学生⑤)

5.2. プログラミング学習 AI「Progate」

<https://prog-8.com/>

「Progate」は、プログラミングを独習するための AI で、iPhone とアンドロイドに対応し無料である。学習できる言語は HTML & CSS, JavaScript, Ruby など 8 種類である。ゲーム感覚で学べるため、自分に合った言語を探すことが出来る。まず解説があり、指示通りに穴埋めをする。正しければ、次へ進める仕組みである。簡単に画面もシンプルであるため、知識がゼロでも学習できる。キャラクターの「ひつじ仙人」が楽しくコメントしたり、教えてくれたりするので、文系学生でも進められる。プログラミングは右の画面で穴埋めの形で回答する。左の画面で、実際の効果がレビュー出来るため、ミスをしていると指示通りの表示にならない。HTML の場合、見出し作成から始まり、改行したり段落を作るなど、基本的な表示方法から学べる。

【「Progate」の感想（石川）】

ダウンロード段階でアプリのアイコンが可愛く、開いてからも愛くるしいキャラクターが存在、プログラミングっぽくない（堅苦しさ、気難しさがない）印象を受けた（図 15）。ログイン方法も簡単で慣れているスマホでプログラミングを学べる手軽さを感じた。



図 15 Progate キャラクター

Java のレッスン I を受けた。始めにスライドを読み理解を深めてから、演習問題が表示されコードの穴埋めを行なった（図 16）（図 17）。スライドは事前に見て理解を深めたり、演習問題中にスライドに戻ったり、レッスン終了後にも開くことが可能であり、予習復習することが十分に出来た。1 チャプターを終える時間は 1 分以内で、演習問題自体は 1 問 30 秒程度で終わることが出来

た。苦勞することなくチャプターを進めることが出来たため、簡単であった。コードを実際に書き込まず、既に用意されたコードのブロックをはめてステップをクリアしたため、「Duolingo」に似



図 16 スライド説明

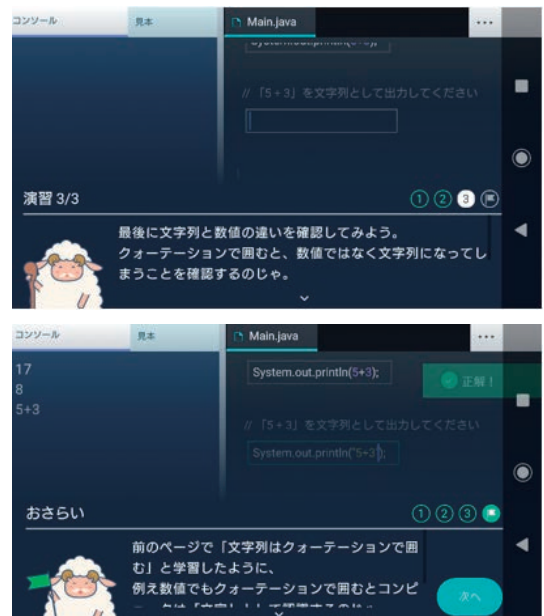


図 17 コード穴埋め

た要素やプログラミングを学ぶ小学生が良く使う「Scratch」⁽¹³⁾のコーディング版のように感じた。

「Progate」はプログラミング（コーディング）のイメージを持つきっかけや、体験を得る目的に適している。プログラミング無知識・未経験の文系学生も、事前準備する必要がなく、スマホとアプリダウンロードをするだけで、羊アイコンを押せば何をすべきか教えてくれるため、簡単にプログラミングを体験できる。簡易性と手軽さから、文系学生がプログラミングを知るきっかけとしてオススメ出来る。

無料で出来る学習環境を提供してくれるAIとして、「Progate」と「Paiza ラーニング」を表3の通り比較する。「Paiza ラーニング」の方が多くのチャプター（分野）を体験でき、解説を見ながら多くコードの書き込み練習が出来るため、十分に基礎を身に付けられる。

【実践環境】

デバイス環境：Android

使用言語：Java（内定企業が主に使う言語であるため選択）

Javaのレッスン：合計4レッスン（うちJava II～IVのレッスンは有料）

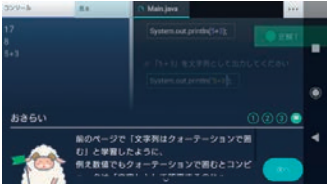
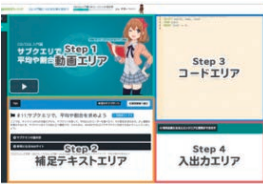
【アプリダウンロード後の画面操作（横向き）】

1. Google アカウントでログイン



図 18 ログイン画面

表 3 「Progate」と「Paiza ラーニング」の比較

	Progate	Paiza ラーニング
使用デバイス	スマホ	PC
ログイン方法	Google アカウントで簡単◎	Google アカウントで簡単 但し、初回会員登録が面倒△
レッスン (Java の場合)	レッスン I 「無料」 レッスン II 以降「有料」	体験版「無料」 入門編 26 レッスン「有料」
レッスン内容	レッスン I 全 14 チャプター 「文字列」「数値」「計算」「連結」「変数の定義～更新」	体験版 全 16 チャプター 「文字列の標準出力・入力」「プログラミングにおいてよくあるミス」「コメント」「数値」「文字列」「四則演算」「変数」「標準入力」「標準出力」「条件分岐」「比較演算子」「ループ処理」
学習方法	順番にステップを踏んでいく。 正解するまで終わらない。 Step1 スライドを読む Step2 演習問題 Step3 コード穴埋め（ブロック選択） 	1つの画面で4つの機能が使える 模範回答がみれない。（有料） Step1 動画視聴 Step2 補足テキスト Step3 コードエリア（入力実践） Step4 出力エリア 

2. Welcome メッセージが表示。「はじめる」をタッチ



図 19 メッセージ画面

3. 画面右にアイコンが4つ、上から「学べる言語」「言語ごとの学習スライド」「レッスン状況」「アカウントについて」を表示する。Web とアプリのレッスン状況が確認できるため、アプリと Web のアカウントを併用して管理できる。



図 20 レッスン状況

文系学生向けツールということであったが、正直、2年間に亘ってゼミで AI を勉強した 4 年生としては、コーディングの達成感に「物足りなさ」を感じました。逆に、プログラミングを経験していなかったり、どう勉強して良いか悩んでいた文系学生には、「ゲームで楽しめる簡易性」と「勉強しやすい手軽さ」を有するため、1 年生や 2 年生がプログラミングを知るきっかけとして、オススメ出来ると思いました。

5.3. プログラミングまとめ

「GitHub」を運営する米ギットハブ CEO のトーマス・ドムケ（2023）は、「学生がコードの自動生成ツール『Github Copilot』をプログラミングの教師として使っている。コンピューターサイエンスを専攻していない学生が機械学習を学ぼう

とする場合。プログラミングの知識がなく苦勞しがちですが、エディターに Copilot を組み込むことで知識のギャップを克服し、解決すべき問題に集中できます」「現時点では、AI が書いたコードを人間がすべてレビューする必要があります。サービスの運用、デプロイ、デバッグなど、我々が DevOps と呼んでいる活動には、現在の AI を超える知識が必要とされるためである。ただ長期的な未来においては、AI がこれらの作業をすべてサポートできるようになる。そして、どんな人でも AI の力を借りれば開発者になれる」と強調した⁽¹⁴⁾。これまでのプログラミング技能は「読む、書く」の 2 つで測定されて来たが、AI を活用することが新たな尺度となり得る。生成 AI が加わり、問われるのは、学生が自らの責任で採用した表現であり、それをコンピューターが自動生成したかどうかは関係ない。高いレベルの表現力や運用力で情報が発信できたかどうかが問われるようになった。

6. 最後に

最近の変化のスピードを 3 倍にしているのは「生成 AI」である。ネット上の膨大なデータがディープラーニングされ続ける世界では、人間を遙かに凌駕する速度でアルゴリズムが進化する。ChatGPT を初めとする新たなテクノロジーの波の到来により、人々や社会が ChatGPT 前に戻ることはない。このような不可逆な変化を前提とした時代に生きるためには、常に足元を見ながら柔軟に調整して進む「しなやかさ」が個人や大学、企業で重要になる。

実際に生成を使ってプログラミングする実習を行うことで、生成 AI のポテンシャルや欠点が見えて来ると考えた。何れにしても、プログラミング生成 AI の成否は、実際にプログラミングを生成してみた人にしか分からないというのが結論である。私たちが可能性を持つ、新しい技術の誕生を目の当たりにしていることは間違いない。大学や企業は、データサイエンス教育やデジタルトランスフォーメーション（DX）導入を数学が分か

らない学生・社員に押し付けカオス状態にしてしまった。久保田（2023）は「数理・データサイエンス教育は研究開発人材の育成を狙いにした色合いが濃い」と批判する⁽¹⁵⁾。一方、生成 AI はプログラミングを民主化する存在である。AI の手助けによって文系学生でも平均的な品質のコンテンツを生み出せるようになった。より熟練した専門家と競争することも可能である。AI は多くの職業で参入障壁を低くしている⁽¹⁶⁾。

2022 年、「ChatGPT」初版が公開されて以降、賛否両論、今後の社会への悪影響に対する批判や反響を呼んでいる。文系学生でも ChatGPT を活用すればコーディングを行うことは出来るが、ChatGPT の活用には、人任せならぬ、「AI 任せにしない」人間との「共存」の形が必要であると分かった。また、文系学生の場合、ChatGPT でコーディング出来たとしても、コードの正誤の判断を行うことが出来ないため、プログラミングが成功するとまで明言することは出来ない。しかし、「ChatGPT を用いれば、プログラミング言語を全く知らない文系学生でも、自然言語処理（NPL）の力を活用してコーディングできる。」という仮説は十分に立証できた。

プログラミングにおける ChatGPT の活用は、どんなに ChatGPT でコーディングが生成できたとしても、プログラミング言語を知らない文系学生にとって生成されたコードのファクトチェックが行えないため、ChatGPT におけるコーディング生成の利便性よりも、既存のコードの誤りや改善点を探すための手段、プログラミング言語を学習するための手段として活用した方が、人間の作業の効率および利便性を高めることが可能である。正確性では課題が残るが、生産性、効率性では人間をはるかに上回る。足元では生成 AI を活用してコーディングの不具合などを調べるソフトも既に登場している。人の手作業を大幅に削減でき、特に初心者には分かり易いコーディングを示してくれるため、プログラミング教育で既存のテキストを用いず、チャット GPT の生成内容を教員が修正した形でコーディングを教えるやり方が最も有用な教育方法となり得る。実際の仕事にお

いても、このような「人間と AI の協働」が仕事の生産性を大幅に上昇してくれることになるであろう。生成 AI のポテンシャルとしては、圧倒的な効率化が出来る点がアドバンテージとして挙げられる。学生は自然言語で動く生成 AI を用いてどんどんコーディングしてみることである。また、ChatGPT によるコーディング生成の実践により、人間の主観が入らない結果から、客観性を持って未学習・未経験だった「コーディング」を捉えることが出来る。学生にとっては、参考書だけでは、知り理解できなかった「ChatGPT」の特性や弱み、プログラミング言語の各性質まで幅広く知ることが出来、非常に有意義な実習になったと評価する。

米バージニア大学のアントン・コリネク氏は研究する際、大規模言語モデル（LLM）がどのような分野で役に立つか、2023 年時点の評価を示した（表 4）。6 つの分野ですべての項目で最も高い評点が付いた分野は、ブレインストーミングなどのアイデア出しと文章校正などの文書作成であった。一方。研究の裏方的業務である文献調べなどについては、最も低い評価に留まった。

表 4 生成 AI と有用性（2023 年時点）
Table 1: Summary of LLM capabilities and rating of usefulness as of Feb 1, 2023

Category (カテゴリー)	Usefulness (タスク)	Usefulness (評点)
Ideation (アイデア出し)	Brainstorming（ブレインストーミング）	3
	Evaluating ideas（アイデアの評価）	2
	Providing counterarguments（反論の提供）	3
Writing (文章作成)	Synthesizing text（箇条書きからの文章作成）	3
	Editing text（文章の編集）	3
	Evaluating text（文章の評価）	3
	Generating catchy titles and headlines (キャッチーなタイトル / 見出しの作成)	3

	Generating tweets to promote a paper (論文を宣伝するためのSNS記事の作成)	3
Background Research (研究の裏方的仕事)	Summarizing Text (文章の要約)	3
	Literature Research (文献調べ)	1
	Formatting References (参考文献の書式設定)	3
	Translating Text (文章の翻訳)	3
	Explaining concepts (概念の説明)	2
Coding (コード関係)	Writing code (コードを書く)	2
	Explaining code (コードの説明)	2
	Translating code (コードの翻訳)	3
	Debugging code (コードのバグ処理)	2
Data Analysis (データ分析)	Extracting data from text (文章からのデータの抽出)	3
	Reformatting data (データの再フォーマット)	3
	Classifying and scoring text (文章の分類と評価)	2
	Extracting sentiment (文章からの感情の抽出)	2
	Simulating human subjects (人間を被験者とするサーベイ)	2
Math (数学的導出)	Setting up models (モデルのセットアップ)	2
	Deriving equations (方程式の導出)	1
	Explaining models (モデルの説明)	1

The third column reports my subjective rating of LLM capabilities as of Feb. a, 2023: (LLM 機能と有用性の暫定的評価)

1=experimental: results are in consistent and require significant human oversight (1 実験的: 結果は整合的でなく人間のチェックが必要)

2=usefulness: requires oversight but will likely save your time (2 有用: チェックは必要だが、時間節約が可能かもしれない)

3=highly usefulness: incorporating these into your workflow will save you time

(3 かなり有用: ワークフローへの導入で時間節約)

出所: Anton Korinek (2023) "Language Models and Cognitive Automation for Economic Research", <https://www.nber.org/papers/w30957>

7. 今後の展望～コード生成 AI の先駆と拡張

日本のデジタル化が欧米やアジアの新興国に比べ遅れつつある事象が顕著になりつつある。IT オンチや情弱（情報弱者）は高齢者に限った話ではない。プログラミングの必修化やデジタル教科書の導入に伴い、デジタル技術を使いこなせる子供とそうでない子供の間にデジタル格差の解消は学校における課題の1つになっている。大学生でも課題をグーグル「クラスルーム」できちんと提出できずに頼み込んで来る学生が結構いる。大学でもハンコを必要とする紙の書類⁽¹⁷⁾が消滅しない原因は対応できない教職員が未だいるからだとの説明を受ける。マイナ保険証の導入が延期されたのは低リテラシー層への配慮であった。デジタル庁は「誰一人取り残さない、人に優しいデジタル化」をミッションに掲げる。デジタル格差を解消できると期待されているのが対話型 AI の ChatGPT に代表される生成 AI の活用である。デジタル格差の解消を考えた場合、デジタルに慣れていない人ほどユーザーインターフェース (UI) として自然言語でのやり取りが向いている。ChatGPT はかなり曖昧な表現でもかなりの精度で解釈してくれる。生成 AI であれば、自然言語でコンピューターとやり取り出来る。

自然言語でやり取り出来る大規模言語モデル (LLM) を学生が活用する際の「PDCA」は次に挙げる3点から容易に理解できる。第1に、LLM の操作のためにプログラミングを勉強する必要はない。自然言語で指示するだけでアウトプットを返してくれる LLM は特殊なスキルを持ったデータサイエンティストでなくても PDCA を回し易くなった。生成 AI 登場以前に計画されたデータサイエンス系の学部・学科・研究者の焦りが出ている背景である。計画したこれらの人たちはアップデート出来ていない。第2はコストが安い点で

ある。インフラやAPI利用料にはコストが掛かるが、ゼロから特定用途に特化した深層学習モデルを構築するためのデータの収集や前処理、学習に必要なコストと比較すれば、GPTといったプラットフォームが提供する基盤モデルを使い、ドメイン知識として少量のデータだけで使えるという点は金銭的な面でメリットが大きい。第3は用意するデータが少量で良い点である。大量のデータを自前で用意しなければならない従来のモデル構築と比べて、LLMは用意するデータが圧倒的に少ない。仮にターゲットとする研究分野の現場が有するデータ量が乏しかったとしても、LLM活用の絶対的な障壁にはならない⁽¹⁸⁾。

イアン・ブレマー（2023）は「AIは民主化された技術だ。ほぼ最先端のAIモデルや、数か月前に利用可能になったアルゴリズムよりも強力なモデルを簡単なプログラミング技術とパソコンさえあれば誰でも使える」と指摘する⁽¹⁹⁾。生成AIの登場で文系学生がAIエンジニアになるハードルはかなり下がった。ChatGPTはプログラミングの良い教師であり、文系学生がエンジニアを目指すのも楽になった。生成AIは自然言語で操作できる分、技術に明るくない学生も理解し易い。生成AIはある意味、「貧者のツール」である。知識や経験、お金がない文系学生を大いに助けてくれる。最新技術を使いこなすことは難しいが、学生は積極的に実習してくれた。

私たちは電卓が使えても算数を学ぶ。電卓の普及により難解な計算を自力で行う必要はなくなったが、数学の素養がなければ電卓の高度な機能も宝の持ち腐れになってしまう。同様にAIが全てを代替する「未来」は考えにくく、人間のプログラミング教育は続くだろう。ChatGPTで自分が欲しい答えを得たいと思えば、適切な質問を投げかけないと返ってこない。『問う能力』も更に必要になっており、自分がなぜ、何に対する回答を欲しているのかを明確にする力も、ますます重要になって来る。

全てのテクノロジーにはリスクと恩恵がある。世界にとって、この技術は利点の方がはるかに大きい。生成AIに対して、誤回答や雇用への影響、

著作権やセキュリティをめぐる問題、米IT大手へのデータや計算能力への集中への懸念などを強調して批判するテクノフォビアも多い。人の作業をAIが代替することが増えるであろう。ただ人間がやることがなくなるかと言えば、そうではない。現在はAIと人間の役割分担のバランスを探っている段階と言える。ChatGPTが登場したころは文章を書くためのアシスタントだったが、今はプログラミングを生成してくれる存在になった。何時間も掛かる大変な作業を自動で出来るようになった⁽²⁰⁾。生成AIの登場で特別な技術がなくてもプログラミングが出来るようになり、今後ますますプログラミング分野でAIの使用が見込まれる。社会人から大学生、高校生まで誰もがデータを持つようになり、生成AIを使えば、独自のAIを作る垣根は下がる。生成AIの普及により、誰もがデジタル技術の恩恵を得られる世の中の到来を願ってやまない。ここまで、プログラミング生成を技術や演出という観点で、専門ゼミナールや演習実習における現場からアイデアや事例を紹介した。今後のAI教育が行われる上で、これらの現場の実践的な知見がヒントになれば幸いである。

8. まとめ

コロナ禍の3年間、日本だけがマスクをしては～っとしている間に世界は劇的に動いていた。GPTを基盤とした生成系AIの開発により、人工知能のトレンドはデータサイエンスのような「予測系」から「生成系」に移った（表5、表6、図21）が、大学でのAI教育と言うと、今でも「予測系」の「データサイエンティスト育成」にあると考えている人も少なくない。コロナ禍前の識別系、予測系などのAI技術は「レッドオーシャン（過当競争市場）」になった⁽²¹⁾。

データサイエンスやデジタルトランスフォーメーション（DX）など高度な理系知識を必要とする教育は文系学部では数学や統計学の基礎学力やノウハウがないためカオス状態を招いた⁽²⁵⁾。数学を入試で課さない大学がデータサイエンス教育

するのは泳げない子供にスキューバダイビングを教えるようなものである。大半は溺れることが必至である。5%の「上位層」はデータサイエンスを学ぶ必要があるが、95%の「中間層」はAIを使いこなす方に時間を割いた方が良い。久保田（2023）は「数理・データサイエンス教育は研究開発人材の育成を狙いにした色合いが濃い」と批判する⁽²⁶⁾。生成AIが注目される影響で、データサイエンスやDXなどは早くも「冬の時代」を迎えている⁽²⁷⁾。DXが上手く行かなかった⁽²⁸⁾のは、コンピューターやソフトの導入だけで済まそうとして、従来のワークフローを徹底改革する覚悟がなかったからである。DXで企業や組織を変えるためには制度や人の行動様式に関する人文・社会科学の知見が要る。コロナ給付金やCOCOA（新型コロナウイルス接触確認アプリ）の混乱を見れば分かる通り付け焼刃では行かない。マイナンバーカードでも失敗を重ねた。文理を癒合した「総合知」が求められる。

コード生成AIを用いた実習のポイントは「創作の楽しさ」が学生の学習に対するモチベーション社会にもたらす効果である。自然言語でコンピューターを動かせる生成AIの登場で、文系学生が楽しみつつ何かオリジナルの創作物が出来る実感を持てるようになった。新技術やイノベーションに否定的なテクノフォビアが多い中、学生たちは新しい技術を使うイノベティブな姿勢を見せてくれた。オリジナリティ、クリエイティビティ、セレンディピティと言っても、結局のところ、読書や経験、既存の表現などの順列・組合せを変えて変奏しているに過ぎなかった。それをAIが模倣できるようになったということである。人間にしか出来ないことはどんどん狭められている。教育現場も生成AIが簡単に書いてしまう「〇〇について論ぜよ」「感想文を書け」みたいな課題を学生に課すことは有効でなくなっている。

プログラミングと聞くと「難しそう」と尻込みする文系学生も生成AIを上手く活用すれば、自分がそれまで出来なかった分野にもチャレンジ出来る。「良く分からない」と思わず挑戦してみる。学生には生成AIを使って新しいことにどんどん

表5 AIの4タイプ

AIの4タイプ分類

1	識別系	見る力	画像認識AI、顔認識AI、文字認識AIなど
2	予測系	予測する力	需要予測AI、顧客予測AI、異常検知AIなど
3	会話系	話す力	自然言語処理AI、翻訳AI、音声認識AIなど
4	実行系	動く・作る力	自動運転AI、ロボット/ドローンAI生成AI、作曲AIなど

出所：野口（2023）⁽²²⁾を植田が修正

表6 AIの変遷

会話系・言語系の技術飛躍
作るAIの飛躍

1	識別系	見る力	異常検知、認証、AIカメラ、OCR
2	予測系	予測する力	需要予測、顧客予測 パーソナライズ、最適化
3	会話系	話す力	チャットボット、SNS分析
4	実行系	動く・作る力	自動ロボ、パナー、顔合成

出所：野口（2023）⁽²³⁾を植田が修正

「AI 1.0」から「AI 2.0」へ
「データサイエンス」教育が早くもレガシー化

- 「AI 1.0」では、個別にデータを収集してAIを作り込む必要があった。一つ一つのアプリケーションを作るのにコストが掛かり、利益率が低くなる課題があった。
- 「AI 2.0」の主役である大規模言語モデルは世界中のデータで訓練した一つの巨大な『脳』様々なタスクに対応でき、自動対話や動画生成などに使いたい場合、微調整するだけでいい。




図21 AIの変遷

出所：李（2023）⁽²⁴⁾を植田が修正

チャレンジして欲しい。効率的に仕事を出来れば、その分空いた時間を好きな時間（推し行為など）に有効活用できる。最新テクノロジーを味方に付けられれば、「人間*AI」の相乗効果で自力では届かなかった未来に辿り着けるはずである。生成AIの実習の最大の醍醐味は、第一人者になるチャンスがあることである。既存の製品やサービスの場合、経験豊かなベテランや知識豊富な先

生に追い付くのは難しい。しかし新たな領域であれば、学生でも勝負に参加できる。マニュアルや教科書が未整備で参考出来るノウハウがない最新テクノロジーの場合、自分でトライ＆エラーを繰り返して知識を身に着けなければならない。苦労はたくさんあるが、きちんと理解できれば、「大学や会社で一番詳しい人」になれる。「このサービスのことなら〇〇さんに聞けば大丈夫だ」と社内外から頼られる存在になれる。

謝辞

グーグル「Teachable Machine」を使った独自AI製作については、2023年10月23日（金）「実習C」で、ユーザーローカルの雨宮俊貴様からご講義戴いた。この場を借りてお礼申し上げたい。本稿の作成に際して、2023年度「卒業研究」「専門ゼミナール」「演習C」「実習C」受講学生の協力を得た。この場を借りてお礼申し上げたい。但し、本稿に関する誤りは筆者に帰属している。

参考ウェブサイト

- [1] 日経サイエンス 5月号 <https://www.nikkei-science.com/page/magazine/202305.html>
- [2] 文部科学省「小学校プログラミング教育の手引」
https://www.mext.go.jp/a_menu/shotou/zyouhou/detail/1403162.htm
- [3] RedMonk「プログラミング言語ランキング: 2023年1月」
<https://redmonk.com/sograde/2023/05/16/language-rankings-1-23/>
- [4] geechs job「5分で分かるガベージコレクションの仕組み」
<https://geechs-job.com/tips/details/35>
- [5] 経済産業省「IT人材需要の変化①」
https://www.meti.go.jp/shingikai/economy/daiyosangyo_skill/pdf/001_s03_00.pdf
- [6] Ledge.ai「転移学習とは | メリット・デメリット・ファインチューニングの意味」
<https://ledge.ai/transfer-learning/>
- [7] AI Academy「Pythonプログラミング 基礎文法 速習」
<https://aiacademy.jp/texts/show/?id=73>
- [8] Amazon「12歳からはじめるゼロからのPythonゲームプログラミング教室（サンプル）」
<https://www.amazon.co.jp/exec/obidos/ASIN/B0758BJWV6/otafukulab-22/>
- [9] プログラミング言語「Java」の記事まとめ
<https://www.sejuku.net/blog/category/programming-language/java>

- [10] AI・機械学習のプログラミング言語「Python」の記事まとめ
<https://www.sejuku.net/blog/category/programming-language/python>
- [11] プログラミング言語「JavaScript」の記事まとめ
<https://www.sejuku.net/blog/category/programming-language/javascript>
- [12] Acro vision「コーディングとは? 基本知識4つ | 必要なスキルや注意点を解説!」
<https://www.acrovision.jp/career/?p=2848>
- [13] Engineer Labo「【難易度順】プログラミング言語を一覧で一挙紹介!」
<https://x-tech.pasona.co.jp/media/detail.html?p=2505>
- [14] <https://scratch.mit.edu/>

参考文献

- [1] A.Bundy 長尾真監訳 (1981)『人工知能入門』近代科学社
- [2] Srin Devadas 黒川利明訳 (2018)『問題解決のPythonプログラミング—数学パズルで鍛えるアルゴリズム的思考』オライリー・ジャパン
- [3] トーマス・ドムケ (2023), 日経クロステック (2023)『ChatGPT 産業革命』pp.124-125
- [4] 新谷虎松 (2014)『改訂 Javaによる知能プログラミング入門』コロナ社
- [5] 田中賢一郎 (2019)『ゲームで学ぶ「JavaScript入門」』インプレス
- [6] 高森康雄 (2018)「人工知能の最新応用技術」『Newton 別冊 ゼロからわかる人工知能 基本的な仕組みから応用例、そして未来まで』p48-49. ニュートンプレス
- [7] 古川渉一 酒井麻里子 (2023)『先読みIT×ビジネス講座 ChatGPT』インプレス

《注》

- (1) 清水亮 (2023), 「DIME (2023.6)」47p.
- (2) 2023年5月18日付朝日新聞 2面
- (3) Acro vision「コーディングとは? 基本知識4つ | 必要なスキルや注意点を解説!」
<https://www.acrovision.jp/career/?p=2848>
- (4) 文部科学省「小学校プログラミング教育の手引」
https://www.mext.go.jp/a_menu/shotou/zyouhou/detail/1403162.htm
- (5) デジタルレシビ「Generative AIの企業における活用方法の最新事例ご紹介」(2023年2月21日)
- (6) <https://japan.cnet.com/article/35206064/>
- (7) メソッドとは、オブジェクト指向プログラミングにおいて、処理や動作を実行するためのコードのブロックのこと。特定のタスクを実行するため命令をカプセル化するため使用される。
- (8) 田中賢一郎 (2019)『ゲームで学ぶ「JavaScript入門」』インプレス.p120-121
- (9) Srin Devadas 黒川利明訳 (2018)『問題解決のPythonプログラミング—数学パズルで鍛えるアルゴリズム的思考』オライリー・ジャパン.p125-128
- (10) 新谷虎松 (2014)『改訂 Javaによる知能プログラミ

- ング入門」コロナ社.p16-21
- (11) 古川渉一 酒井麻里子 (2023)『先読み IT × ビジネス講座 ChatGPT』インプレス.p37
 - (12) 日経クロストrend「ChatGPT& 生成 AI 最強の仕事術」117p.
 - (13) Scratch <https://scratch.mit.edu/>
 - (14) トーマス・ドムケ (2023), 日経クロステック (2023)『ChatGPT 産業革命』pp.124-125
 - (15) 久保田啓介 (2023), 2023 年 7 月 2 日付日本経済新聞 26 面
 - (16) AI ツールが技術的でない指示をもっと理解できるようにすれば, 未経験者でもスマホ用アプリや業務用のパソコン用プログラムを書けるようになるだろう。「AI はソフトウェア開発者になるハードルを低くした」。プロの開ソフトウェアだけでは手に負えないような高度なソフトウェア開発を担うようになる。「NEWSWEEK (2023.6.20)」44p.
 - (17) 情報が紙で管理されると, データを取り込むのに非常に時間が掛かり, 今回の生成 AI についても, データを使って事後学習 (ファインチューニング) する必要がある, 結局はデジタル化が進まない。
 - (18) 辻中仁士 (2023), 2023 年 11 月 2 日付日経産業新聞 14 面
 - (19) イアン・ブレマー (2023), 2023 年 7 月 6 日付日本経済新聞 7 面
 - (20) AI ツールを人間と一緒に活用した場合にコーディングに必要な時間は, 人間だけの時に比べて 55% も短くなると言う。「NEWSWEEK (2023.6.20)」44p.
 - (21) データサイエンス学部・学科が急増し, レッドオーシャン (過当競争市場) となった。数学の素養ある高校生は増えておらず, 未熟な学生を生んでしまった。
 - (22) 野口竜司 (2023)「シンギュラリティは近い?! 次世代 AI の今とこれから」
 - (23) 野口竜司 (2023)「シンギュラリティは近い?! 次世代 AI の今とこれから」
 - (24) 李開復 (2023), 2023 年 4 月 21 日付日本経済新聞 18 面
 - (25) 数学を教えないままに, 数学の素養がない文系学生を集めて「データサイエンス教育」や「デジタルトランスフォーメーション (DX) 教育」を謳うのは, 「デジタル後進国」と揶揄された政府と安易な大学の「共犯」と言える。実際に学ぶ学生の能力や実務を担う先生の状況を十分確かめずデジタル教育の運用を始めたことがカオス状況を引き起こしたのは明らかである。カオスになった反省から, 文系学生へのデータサイエンス教育や文系社員への DX 推進が難しいことが判明したが, 数学力を伸ばすのに必要であるのは, 高校以上の指導改善である。日本の高校は大学受験に対応するため 2 年生で生徒を文系理系に分け, 文系は数学をあまり学ばなくても卒業できるようにしている。桜美林大学の芳沢光雄名誉教授は「米国や英国などは数学は全ての科学技術の発展に不可欠という考えが根付いており, 文理の明確な区分はない」と指摘する。2023 年 7 月 8 日付日本経済新聞 38 面。大学の現場からは「理系人材の育成は必要だが, 単なる看板の掛け替えに終わるのではないかという懸念もある」という指摘が出ている。2023 年 7 月 23 日付朝日新聞 3 面
 - (26) 久保田啓介 (2023), 2023 年 7 月 2 日付日本経済新聞 26 面
 - (27) データサイエンスに数学は欠かせない。予備校「河合塾」が私立大学のデータサイエンス系学部・学科の一般選抜を調べたところ, 募集区分の約半数が「数学なしで受験可」だった。河合塾は「数学の素養を身に着けた受験生は増えていない」と言う。日本の高校では大学入試に対応した文・理系クラス分けが当たり前である。2023 年にデータサイエンス学部を開設した京都女子大学は高校で習っていない文系学生に向け「入学後に数学を丁寧にサポートする」を掲げている。入学すると 1 年生に対し, 微分・積分からスタートし線形代数の基礎まで教える。「数学Ⅲまで試験科目」とすると受験生が集まらないため, 文系学生にも受験してもらいたいというのが本音だった。2023 年 8 月 3 日付読売新聞 10 面
 - (28) 2023 年に発行された「DX 白書 2023」(独立行政法人情報処理推進機構)の副題は, 「進み始めた『デジタル』, 進まない『トランスフォーメーション』」である。2018 年の「DX レポート」(経済産業省)に端を発する DX ブームから 5 年も経たのに, 多くの日本企業はデジタル活用が部分的な効率化にとどまり, ビジネス変革や組織改革までつなげていない。原因の一つとして, 数学を十分に理解できない文系の経営事務部門と理系の IT 部門とのコミュニケーションが不十分であることが挙げられる。大学の文系学部のデータサイエンス教育や DX 教育の失敗も同様である。注目されているのは, チャット GPT を中心に, 第 4 次 AI ブームを牽引する生成 AI である。生成 AI は, プログラミング言語ではなく自然言語という指示形式で動くため, 数学が分からない文系でも動かせる特徴を持つ。そのため, 「AI の民主化」を加速させ, AI 活用のハードルを極端に下げた。これは, 組織全体でデジタル化の可能性を模索し, ビジネス変革・組織改革を出来る好機となり得る。2023 年 8 月 2 日付日本経済新聞夕刊 7 面

