6R-08

GPT プログラミングの現在地 - 大学受験 AI チューターの開発とリスクの考察

大久保 慶太郎†

桐蔭学園高等学校 2 年†

「Let's think step by step」とは大規模言語モデル（LLM）で AI に思考の連鎖をもたらし性能を高めるマジックワードである。当該論文[1]の著者は当時博士課程 3 年の日本人であり、私は思考過程の言語化に興味を抱いた。その発展途上である自然言語処理（NLP）を用いたコーディングの自動化やチャットボットによるアシスタント機能について、情報戦といわれる大学受験に向けて AI チューターの開発を行いながら研究に取り組んできた成果を発表する。

対象とした私立大学 10 校の入試要項は PDF で提供されており、複数学部出願時における留意点や詳細な受験倍率等は動画で配信されることが多い。また、情報の開示が不定期であるため頻繁にホームページへアクセスしないと機会損失につながるイベントも存在する。基本となる入学者選抜実施要項は文部科学省が毎年 6 月に公表しているが、個別校による情報提供方法に関しては努力義務であり、URL やファイル名称・記述形式等が揃っていないため検索性・可読性の低さが非効率な要因となっている。そのため学生は勉強時間を割き能動的にホームページを巡回することになるが、この作業が自動化されることにより、集中を切らすことなく更には情報格差の是正につながれば本望だ。

コンセプトは生成 AI によるマルチモーダル・音声反訳・要約・日程調整・プログラミング機能の検証をベースに Web アプリケーションを開発し、最新情報をスマートフォンへ PUSH 通知することで前述の課題解決につなげることだ。生成 AI は ChatGPT / Google Bard / BingAI、プログラミングは Visual Studio Code（VSCode）、各プラグインは評価に基づき以下を選定した。

- 開発補助： GitHub Copilot[2] / Codex
- 最新情報の抽出： Webpilot
- 動画の要約： Video Insights
- PDF の要約： Chat with PDF / PyPDF2
- スケジュール作成： Show me Diagram

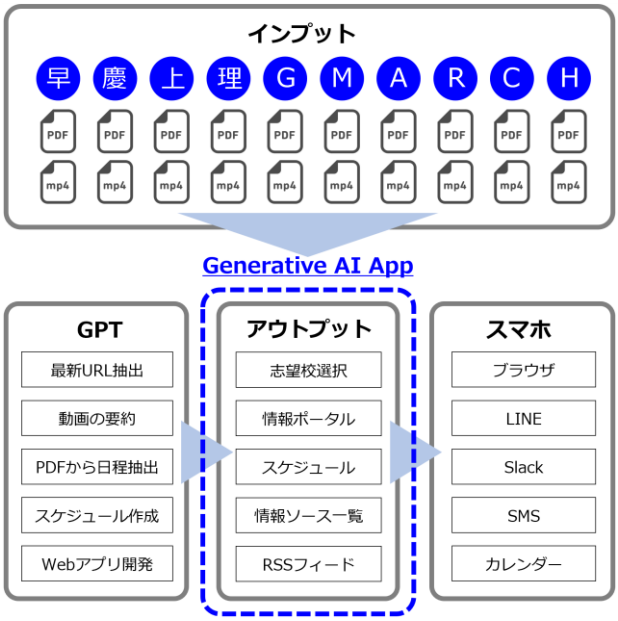

図 1： コンセプト

・最新 URL 抽出： 情報のソースは ①大学のポータルサイト、②入試個別サイト、③外部委託先（52school.com）に掲載されており、URL 構成やファイル名が異なるため同じプロンプトから入試要項の最新 PDF を一括で抽出することはできない。更新される頻度は軽微かつ RSS や SNS 等での検知が可能であり、準備段階ではあらかじめ取得した PDF をローカルフォルダにファイル名称を統一して保管する。

・動画の要約： 前述の①と②に加え、YouTube の各校公式サイトで公表されているが、閲覧に ID / Password の入力を求められるものが存在する。セキュリティポリシーへの抵触を考慮し、Video Insights から事前に得られたテキストデータを格納しておく。利用制限（最長 15 分）はあるものの、時間短縮に最も寄与できる機能だ。

・PDF から日程抽出： 表から情報を抽出することは可能であるが、日付表記が統一されておらず「縦書き」や「複数行に跨る表記」「年月日とスラッシュ、半角と全角、チルダとハイフンの混同」が要因となり、誤った期間を抽出するエラーケース（ハルシネーション）が発生した。

Explore GPT Programming Proficiency - Development & Risk Considerations for Exam Management AI Tutors

†Keitaro OKUBO, Toin Gakuen High School (Sophomores)

これらはデジタル庁の『データ要件・連携要件標準仕様（教育分類）』でも定義されておらず、推奨データセットを用いた整形が必要となる。

・**スケジュール作成**： 前の工程で整備したデータからタスクを抽出し、Mermaid 記法へと自動変換した上でガントチャートを生成する。続いてイベント情報を Python コードに変換し、Google カレンダーの API を介してスマホに登録する。

図２： 生成 AI によるコードベースデザイン

・**Web アプリ開発**： 開発言語は Python / TypeScript / CSS、そして生成 AI は VSCode と組み合わせて GitHub Copilot と Codex を活用した（図３）。UI デザインは DALL-E[3]が有効であり、ランディングページのイメージやナビゲーションバーの生成を想定通り行うことができた。開発工程におけるベストプラクティスは DALL-E が生成した UI デザインをマルチモーダル機能で画像分析し、GPT プログラミングで仕様通りにプロンプトを指示することだ（図２）。初期設定を誤ると Copilot では改修しきれないケースが散見され大幅な手戻りが発生するため、コードベースデザインを踏まえたコード生成が重要だ。

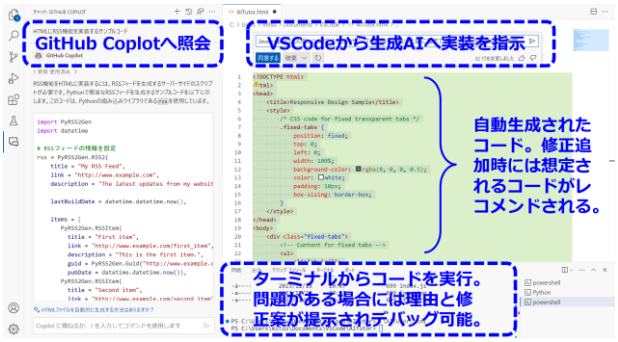

図３： GPT プログラミングのコンソール画面

このように初心者でも生成 AI の支援によりライブラリの追加や環境変数を設定しながらプロト開発を行うことができたが、精度を向上したり各工程で抽出・整形したデータをシームレスに連携するには、図４のような専有環境を用いて LLM の生成プロセスに組み込む必要がある。

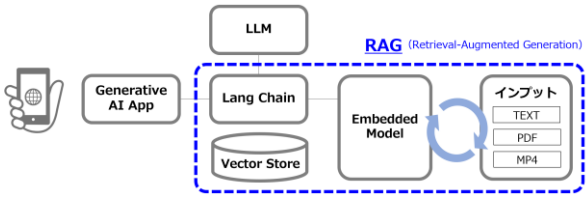

図４： RAG による検索精度の向上

単純にローカルファイルを読み込ませるのであれば OpenAI の Assistants API[4]にアップロードすることで実現できるが、常にデータベースを最新情報に保ちつつ、インタラクティブなアプリケーションの検索性を支える仕組みの一つとして RAG[5]は有効な手段と考えられる。しかし、精度向上における根本的な原因は、各校による情報の提供方法にあると言える。AI に正しく情報を読み込ませるには、JSON フォーマットで記述し、API を公開、仕様は Swagger を参照する仕組みがデファクトスタンダードだ。

本研究ではデータの収集～分析～整形に最も時間を要したが、主旨とは異なる発見もあった。それは大学受験支援企業（パスナビや旺文社）が特定校のホームページにデータ連携用のタグを埋め込んでいる点だ。差別化は LINE や Open Canvas 等で行えばよく、ぜひ各校における協調領域やデータ標準の策定において本研究の成果が役に立つことを願う。

一方、GPT プログラミングにおけるリスク管理の観点では、前述のハルシネーションに加え生成 AI の一般的なリスクと同様、ソースコードに関する著作権の課題が存在する。GitHub Copilot（Business 版）では著作権侵害に抵触する可能性のあるパブリックソースを提案から除外することができ、自身のコードについてもプロンプトに入力した情報の統制が可能だ。また、BingAI では情報ソースの URL を表示する仕組みが実装されておりアウトプットへも織り込んだ。このような運用によってリスクを低減したり脆弱性診断等を併用しながら、これからも情報工学の分野から社会へ貢献していかれるよう、情報格差の是正に向けて取り組んでいきたい。

＜謝辞＞
環境をご提供くださった FINOLAB とメンターのご協力に心より感謝いたします。

＜文献＞
[1] Large Language Models are Zero-Shot Reasoners Takeshi Kojima,etc.
[2] GitHub Copilot · Your AI pair programmer
[3] DALL-E 3
[4] OpenAI Assistants API
[5] Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks