LLM と Docker 環境を統合した 対話型言語処理教育プラットフォームの設計と実装

長谷部 陽一郎
同志社大学
yhasebe@mail.doshisha.ac.jp

概要

LLM と Docker 環境を統合した対話型言語処理教育プラットフォームを提案する．本システムは，Electron ベースのデスクトップアプリケーションとして複数の OS 環境で動作し，Docker コンテナ上に構築された再現可能な言語処理環境を提供する．ユーザは 独自の Web UI や Jupyter Notebook を介して LLM と対話しながら，様々な言語処理ツールを活用した実践的な学習が可能となる．また Ruby を用いた DSL による記述で独自のアプリを作成できるため，多様な教育ニーズに対応した柔軟なカスタマイズが可能である．本プラットフォームは，LLM を用いた新たな教育手法の探求を促進し，言語処理教育の質的向上に貢献すると期待される．

1 はじめに

近年の大規模言語モデル (LLM) の急速な発展は，自然言語処理 (NLP) の様々な分野に大きな変革をもたらしている．とりわけ教育分野においては，LLM を活用した革新的な教育手法が注目を集めており，学習者の個別の習熟度やニーズに対応した学習方法や，インタラクティブな学習環境の構築など，様々な可能性が探求されている[1][2].

しかし，LLM を含む多様な言語処理技術を教育現場で効果的に活用するためには，以下の課題を克服する必要がある．

環境構築の複雑さ：LLM やその他の言語処理技術用いて実際のタスクに取り組む力を身につけるにはには高度なプログラミングスキルや複雑な環境構築が必要である．

ツール管理の煩雑さ：音声認識・合成，画像認識・生成，コード実行などのために，様々な外部ツールやライブラリを導入し管理する必要がある．

カスタマイズ性の不足：各種の LMS（learning management system）や REPL（Read-Eval-Print Loop）環境などは拡張性に欠け，高度な言語処理の方法を学んだり実践したりするためのツールとしては使いにくい

そこで本研究では，これらの課題を解決し，LLM の多様な機能を最大限に活用できる教育プラットフォームとして Monadic Chat を提案する[i]．本システムは，Docker を活用することで，必要なツールやライブラリがプリインストールされた再現性のある実行環境を容易に構築・共有することを可能にする．これにより，Web UI や Jupyter Notebook を介して LLM と対話しながら様々な言語処理ツールを活用した実践的な学習が可能となる［付録 図 A, B, C］．なお，Monadic Chat はオープンソースのソフトウェアとして https://github.com/yohasebe/monadic-chat で公開されている．

2 システム概要

本システムは，学習者が複雑な環境構築やプログラミングに煩わされることなく，LLM の多様な機能を活用した実践的な学習を容易に行えるように設計されている．以下では，システムアーキテクチャと主要な機能について説明する．

2.1 システムアーキテクチャ

図 1 に Monadic Chat のシステムアーキテクチャを示す．Monadic Chat は，Electron ベースのデスクトップアプリケーションとして実装されており，Windows, macOS, Linux 環境で動作する．内部では Docker を利用して，複数のコンテナが連携する形で

[i] Monadic Chat の初期バージョンについては[3]で論じられている．それは，対話を「モナド」として管理していくという概念に基づき，LLM のテキスト補完 API で文脈を保持するためのフレームワークであった．本論文で示すシステムは，これを大幅に拡張したものである．

システムが構成されている．標準で構築されるコンテナは，Ruby コンテナ，Python コンテナ，Selenium コンテナ，PGVector コンテナである．これらのコンテナは Docker Compose によって管理され，ユーザは簡単な UI 操作［付録 図 D］でシステム全体の起動・停止を行うことができる．

2.2 主要な機能

Monadic Chat は，言語処理教育を支援するための様々な機能を提供する．主な機能は以下の通りである．

多様な LLM の利用 OpenAI，Anthropic，Google，Cohere，Mistral AI，xAI などの主要な LLM の API を利用可能である．利用する LLM を自由に選択することで，様々なモデルの特性を比較・検討しながら学習や作業を進めることができる．また Ollama を用い，Docker 環境内にローカル LLM を構築して使用することも可能である[ii]．

Docker による環境構築 Docker を利用することで，必要なツールやライブラリがプリインストールされた再現性のある実行環境を容易に構築可能となる．学習者は環境構築に時間を費やすことなく，言語処理の実践的な学習に集中できる．

多様な対話機能 テキストベースの対話に加え，音声認識・合成，画像認識・生成，コード実行など，LLM の多様な機能を活用したインタラクティブな学習体験を提供する．

アプリケーション開発フレームワーク Ruby による簡易なスクリプト記述で独自のアプリを作成可能．システムプロンプトやパラメータ設定などをカスタマイズすることで，多様な教育ニーズに対応した柔軟なシステム構築が可能となる．

多様な入出力 LLM への入力に際して各種のドキュメントファイルのほか，URL で Web ページを指定可能である． LLM との対話におけるデータの入力と出力は，共有フォルダを用いて行うことができる．また対話データの保存と読み込みにも対応している．

3 実装

Monadic Chat は，多様な LLM との連携，Docker による環境管理，そして柔軟なアプリケーション開発フレームワークを提供するために，複数の技術を組み合わせて実装されている．以下では，Docker コンテナとアプリケーション開発について説明する．

3.1 Docker コンテナ

Monadic Chat は Docker を基盤としており，複数のコンテナが協調動作することでシステム全体の機能を実現する．各コンテナは特定の役割を担い，互いに通信することで複雑な処理を実行する．

Ruby コンテナ システムの中核を担う．LLM との API 通信や Web インターフェースの提供，各コンテナ間の通信管理などを担当する．ユーザが作成した Ruby スクリプトによるアプリもこのコンテナ上で実行される．

Python コンテナ Python 環境を提供し，Jupyter Notebook, Pandas, NumPy, Matplotlib などの主要なデータサイエンスライブラリをプリインストールしている．ユーザは Python スクリプトを実行したり，Jupyter Notebook を利用した対話的なデータ分析を行うことができる．

Selenium コンテナ Selenium と Headless Chrome を用いて Web ブラウザ操作機能を提供する．Web スクレイピングや Web アプリケーションのテストなどに利用可能である

PGvector コンテナ PostgreSQL と PGvector 拡張機能を用いて，テキスト埋め込みベクトルの保存・検索機能を提供する．RAG など，埋め込みベクトルを利用した高度な言語処理タスクを効率的に実行できる．

ユーザは，必要に応じて独自の Dockerfile と docker-compose.yml ファイルを作成することで，新たなコンテナを追加することが可能である．これにより，Monadic Chat の機能を容易に拡張できる．NLTK や spaCy などの言語処理ライブラリや辞書データをあらかじめ設定しておくこともできる．

3.2 アプリケーション開発

Monadic Chat では，Ruby を用いた DSL による簡易なスクリプト記述で独自のアプリを作成できる[iii]．各アプリは，システムプロンプト，LLM のパラメータ設定，関数呼び出しなどを定義することで，LLM の動作をカスタマイズし，特定のタスクに特化した対話型インターフェースを提供する．LLM からは，Ruby のメソッド呼び出しに加え，Python

[ii] https://ollama.com/

[iii] 独自アプリの作成は DSL（domain specific language）として Ruby を用いるが，アプリ内で使用する機能は Python など他言語でも作成可能である．

コードの実行, Jupyter Notebook へのセルの書き込みと実行，Selenium による Web アクセス，PGVector 上の埋め込みベクトルデータへのアクセスなどが可能である．また，画像などのファイル生成を伴う処理の場合，共有フォルダ機能を用いて生成されたファイルにアクセスできる．

例えば，動画ファイルを受け取り，音声抽出，文字起こし，統語解析を行うようなアプリを作成することも用意である．DSL で基本的な仕様を記述し，必要となる内部処理を適宜 Python などの言語で記述することができる．処理結果は，LLM との対話画面に表示するとともに，共有フォルダにも保存されるため，ユーザは必要に応じてアクセスできる．このように，Monadic Chat のアプリケーション開発フレームワークは, Docker コンテナ上のツールやライブラリ，および外部 API をシームレスに連携させることで，柔軟で拡張性の高いアプリ開発を可能にする．

3.3 Jupyter Notebook との連携

Monadic Chat は，Python コンテナ内で Jupyter Notebook を起動する機能を提供し, インタラクティブなデータ分析と学習を支援する. この機能は, Web インターフェース上のメニュー操作または LLM への自然言語での依頼により利用可能になる．

Jupyter Notebook が起動すると，ユーザーは LLM との対話を通じて，テキストセルやコードセルへの書き込み，コードの実行，そして分析結果の解釈といった一連の作業をシームレスに行うことができる．

例えば，「新しいノートブックを作成し『感情分析』というタイトルを設定してください」と指示すれば，LLM は Python コンテナ内の Jupyter Notebook サーバーにリクエストを送信し, 対応する処理を実行する. 同様に,「最初のセルに Markdown 形式で『はじめに』と記述して，簡単な解説文を続けてください」といった指示でテキストセルを追加することも可能である．

コードセルへの Python コードの記述と実行も，LLM との対話を通じて行える．例えば，ユーザーが「IMDB 映画レビューデータセットを用いて，レビューテキストの感情分析を行うコードを記述してください」と指示すると，LLM は適切な Python コードを生成し，コードセルに挿入する．その後，データセットの読み込み，前処理，特徴量抽出，モデル学習，そして評価といった一連の処理を指示することも可能である[iv].

Monadic Chat の Jupyter Notebook 連携機能は，LLM の高度な言語理解能力とコード生成能力を活かすことで，ユーザーがインタラクティブな環境でデータ分析のスキルを習得し，実践的な課題に取り組むことを支援する．これにより，従来のプログラミング教育における環境構築の煩雑さや，コード理解の難しさといった課題を克服し，より効果的な学習体験を提供することが期待される．

3.4 Docker 環境へのアクセス

Monadic Chat は，Docker コンテナへのコマンドラインアクセスを提供することで，高度なユーザーによる環境のカスタマイズを可能にする．Monadic Chat のコンソール画面には, 各コンテナにアクセスするための docker exec コマンドが表示される．ユーザーはこれらのコマンドをコピー&ペーストしてターミナルで実行することで，各コンテナのシェルにログインできる．LLM は各コンテナを構築するために用いた Dockerfile を参照できるため, 自身の初期環境についての知識を有している．

この機能により, 知識のあるユーザーは, LLM と対話しながら Docker 環境を自身のニーズに合わせて調整することが可能となる．例えば，ユーザーが新しいライブラリをインストールしたい場合，LLM に「Python コンテナに transformers ライブラリをインストールするにはどうすれば良いですか？」と質問できる．ユーザーは指示に従ってコマンドを実行することで, Python コンテナに transformers ライブラリをインストールできる[v].

3.5 初期設定のカスタマイズ

Monadic Chat の Docker 環境は，共有フォルダに配置された pysetup.sh スクリプトを用いて，コンテナのビルド時に初期状態でインストールされるツールやライブラリをカスタマイズできる．ユーザーはこのスクリプトを編集することで，必要なライブラリやツールをあらかじめインストールしておき，自身の学習や研究に最適な環境を構築できる．この機

[iv] 当然ながら，生成されるコードや説明文の質や適切さは, 使用するモデルやユーザーからのプロンプトの内容に左右される．

[v] 簡易なものであれば LLM 自体に Docker 環境に各種のツールやライブラリのインストールを依頼することも可能である．

能は，Monadic Chat が実行する Docker のビルドプロセスに組み込まれているため，ユーザーが手動でインストール作業を行う必要がなく，環境構築の手間を大幅に削減できる．

これらのカスタマイズ機能は，Monadic Chat の柔軟性と拡張性を高める重要な要素である．ユーザーは LLM との対話を通じて，あるいは pysetup.sh を用いることで，自身のニーズに合わせた最適な Docker 環境を構築し，様々な言語処理タスクに活用できる．

4 言語処理教育における活用例

4.1 プログラミング学習

Monadic Chat の Jupyter Notebook 連携機能は，インタラクティブなプログラミング学習環境を提供することで，言語処理技術の習得を支援する．学習者は LLM と対話しながら，Python コードを記述・実行し，その結果をリアルタイムで確認できる．さらに，LLM からコードの解説やエラー発生時の原因特定，修正方法の提案といった支援を受けることで，実践的なプログラミングスキルと合わせて，言語処理のアルゴリズムやライブラリの理解を深めることができる．例えば，自然言語処理の初学者にとって，特定のライブラリの使用方法やアルゴリズムの実装は難しい課題となる場合が多い．Monadic Chat を用いることで，学習者は LLM に自然言語で質問し，具体的なコード例や解説を得ながら，形態素解析や構文解析，感情分析といった具体的な言語処理技術を習得できる．

4.2 データ駆動型言語分析

人文系の言語学においても，近年，データ駆動型の言語研究が広く促進されている[4][5]．具体的には，大規模コーパスからの文字データや音声データを用いて，定量的な観点からの分析を行うような研究である．本システムを使うことで，このような分析を，LLM の支援を受けながら効率的に行うことができる．例えば，Selenium コンテナを用いた Web データのスクレイピングにより特定の言語表現の用例を収集したり，PGVector コンテナを用いた類似文書検索を活用して自ら採取した言語データにおける語彙の分布を調べるといった様々な分析タスクを LLM と対話しながら実行できる．人文系の研究者にとって，Python などを使ってデータの分析や可視化を行うことは敷居が高いと感じられることが多い．しかし LLM との対話を通じて，使用すべきコードやその詳細についての理解を深めることで，自然言語処理技術を用いた分析のスキルを段階的に身につけることができる．

4.3 言語教育への応用

Monadic Chat は，音声入力・出力機能やファイル入出力機能を備えているため，外国語学習支援にも活用でき，応用言語学の発展にも貢献する．例えば，学習者は LLM と音声で会話練習を行うことで，発音やイントネーションの矯正，会話表現の習得などが期待できる．テキストファイルを入力として LLM に文法チェックや添削を依頼したり，音声ファイルを出力として発音練習に利用することも，言語習得の支援につながる．また，ユーザーによる LLM との会話は JSON フォーマットで保存・再現が可能であり，学習者の活動記録を把握・管理することも容易である．このように，LLM と Docker の統合による柔軟性と拡張性を活かすことで，学習者の多様なニーズに対応した，効果的な教育環境を構築することができると期待される．

5 おわりに

本稿では，LLM と Docker 環境を統合した対話型言語処理教育プラットフォーム Monadic Chat の設計と実装について述べた．Monadic Chat は，Docker コンテナ上に再現性のある言語処理環境を提供することで，学習者の環境構築の負担を軽減するとともに，LLM の多様な機能を活用したインタラクティブな学習体験を可能にする．さらに，Ruby を用いた DSL によるアプリケーション開発フレームワークは，柔軟なカスタマイズ性と拡張性を提供し，多様な教育ニーズに対応できる．

今後の展望としては，まず，マルチユーザーによるリアルタイム対話機能の実装が挙げられる．これにより，学習者どうし，あるいは学習者と指導者間のインタラクティブなコミュニケーションを促進し，協調的な学習環境を構築することが期待される．また，Monadic Chat を実際の教育現場に導入し，その効果を検証することで，システムの改善や新たな教育手法の開発につなげたい．

謝辞

本研究は JSPS 科研費 24K00078（生成 AI を組み込んだ日本語作文診断システムの開発と普及に関する研究）の助成を受けたものです.

参考文献

1. Enkelejda Kasneci, et. al. 2023. “ChatGPT for good? On opportunities and challenges of large language models for education” *Learning and Individual Differences* 103.
2. Shen Wang, et. al. 2024. “Large Language Models for Education: A Survey and Outlook” arXiv:2403.18105 [cs.CL]
3. 長谷部陽一郎. 2023.「Monadic Chat：テキスト補完 API で文脈を保持するためのフレームワーク」『言語処理学会第 29 回年次大会発表論文集』pp. 3138–3143, 2023.
4. 李在鎬［編］2019.『ICT×日本語教育』ひつじ書房.
5. 中谷健太郎. 2024.『パソコンがあればできる! ことばの実験研究の方法：容認性調査，読文・産出実験からコーパスまで』第 2 版. ひつじ書房.

付録（実行画面）

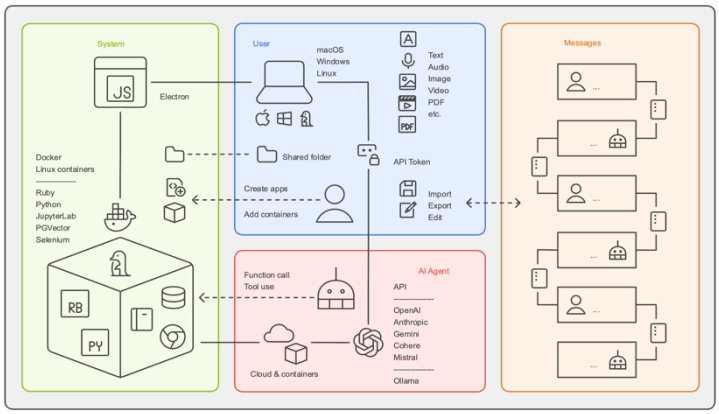

図 1：Monadic Chat の構成

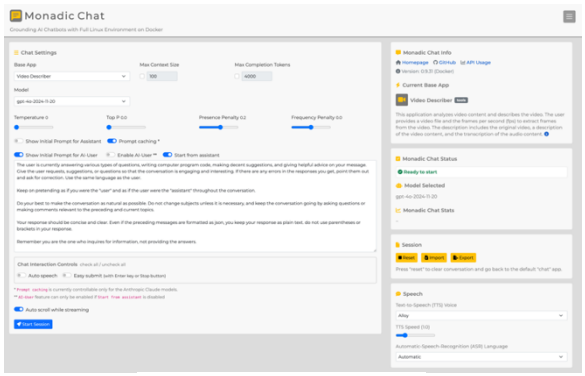

図 A：Web UI メニュー画面

図 B：対話画面の例

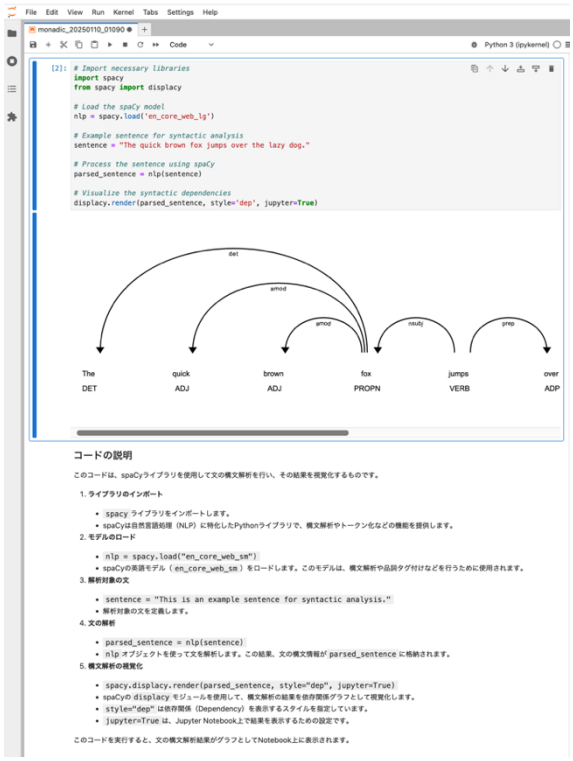

図 C：LLM による Jupyter Notebook への書き込みと実行

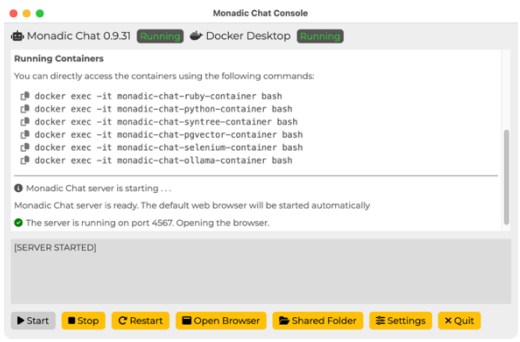

図 D：Docker コンテナ管理画面