1ZB-01

大規模言語モデルに基づく IoT サービスの実現

横辻　龍太郎†　　林　冬惠‡

岡山大学工学部情報系学科†　　岡山大学学術研究院環境生命自然科学学域‡

1. はじめに

近年，ChatGPT や BERT の登場により，大規模言語モデルに対する関心が高まっており，様々な分野での活用が研究されている．大規模言語モデルは，事前に膨大な量のテキストデータによる学習によって文章生成，対話応答などの自然言語に関するタスクを行うことができる．

しかし，大規模言語モデルはリアルタイムな物理世界の情報に基づいてユーザのリクエストに対する回答を生成することはできない．

物理世界の情報を取得する手段として，Internet of Things(IoT)や Web サービスがある．大規模言語モデルが IoT や Web サービスから取得した情報に基づく回答を可能にすることで，大規模言語モデルによる物理世界の制御が可能になる．IoT 分野での大規模言語モデルを用いた研究は，主に，ユーザのリクエストに対する自然言語処理に焦点を当てている［1］.

物理世界の状態の取得，制御が可能な IoT デバイスは，その数や種類を増やし続けており，それに伴って多様なデータ形式が存在する．多様なデータ形式はアプリケーション開発におけるデータの相互運用を困難にする［2］.

本研究の目的は，大規模言語モデルが物理世界の情報に基づく回答を生成できる IoT サービス基盤を実現することである．そのための課題として，IoT サービスの情報に基づく回答生成，生成した回答によるアクチュエータの制御のための IoT サービスと大規模言語モデルの接続，データ形式の異なる IoT の相互運用性の実現がある．

2. 大規模言語モデルと IoT の接続

大規模言語モデルは回答のルールを決めることで，プロンプトに対してルールに沿った回答を生成することができる．大規模言語モデルと IoT の接続によって，物理世界の情報に基づく回答を生成，活用するために，具体的な実世界での利用シナリオに基づく自然言語のルールを定義し，取得した IoT データをプロンプトに変換して大規模言語モデルにリクエストする．この手法によって，大規模言語モデルがセンサ情報に基づく回答，アクチュエータの制御が可能になる．

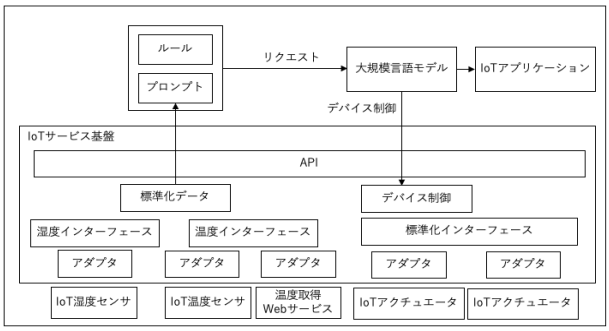

図 1 IoT サービス基盤のアーキテクチャ

3. IoT の相互運用性の実現

図 1 は提案する IoT サービス基盤のアーキテクチャである．データを標準化して扱うために大川らの提案手法をもとに階層的なインターフェースを設計する［3］．温度や湿度，照度などのデータの種類ごとのインターフェースを設計し，そのインターフェースの記述に沿って IoT センサや Web サービスごとのデータの違いを吸収するアダプタを設計する．具体的に，温度データの場合は，センサの抽象インターフェース Sensor クラスを定義する．Sensor クラスを継承する形で，温度を取得するための getTemperature メソッドを持つ温度インターフェース TemperatureSensor クラスを実装する．TemperatureSensor の記述に従って温度センサや温度取得の Web サービスごとのデータ形式に沿った getTemperature を記述する．これによって，アプリケーション開発者は，データの違いを気にすることなくデータを相互運用できる．

4. 実装および評価

図 1 に基づいて，スマートホーム環境での利用シナリオを想定したアプリケーションと IoT サービス基盤を実装した．ユーザはアプリケーションから現在の気温をはじめとした物理世界の情報をリクエストすると，IoT 基盤からデータを取得し，プロンプトに変換する．その後，大規

Realizing IoT Services Based on Large Language Models

Ryutaro Yokotsuji†, Donghui Lin‡

†Department of Information Technology, Faculty of Engineering, Okayama University

‡Faculty of Environmental, Life, Natural Science and Technology, Okayama University

模言語モデルは回答を生成し，ユーザに表示する．

実装した IoT サービス基盤を通してデータを取得した場合，データを直接取得する場合と比べてオーバーヘッドが生じる．実世界での利用シナリオを実装し，ユーザのリクエストから回答表示までの応答時間の平均を計算し，その差を比較することで，実装した IoT サービス基盤によるオーバーヘッドを測定する．IoT 基盤を用いない場合，IoT センサデータはデータベースから直接取得し，Web サービスの場合は直接 API の呼び出しを行うことでデータを取得し，データの取得以外は同様の環境で実装する．

図 2 は大規模言語モデルによる対話生成の時間を除いたユーザのリクエストからの応答時間を検証した結果である．A，C は IoT センサから取得した 1 つのデータを利用したシナリオである．IoT データは，基盤を用いない場合，データベースから直接取得している．オーバーヘッドは約 100ms ほどであり，ユーザの体験に大きな影響を与えるほどではない．

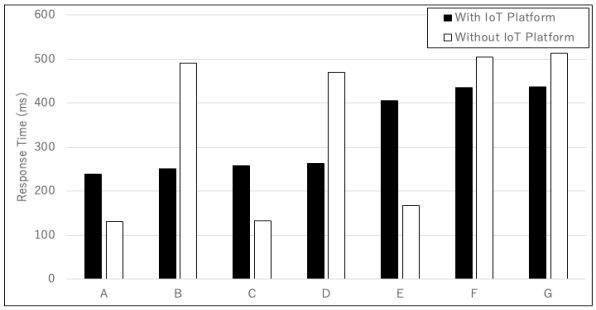

図 2 対話生成を除く平均の応答時間（20 回）

B，D は Web サービスから取得した単一のデータを用いるシナリオである．IoT 基盤を用いない場合，API から直接呼び出すことでデータを取得する．IoT 基盤を用いない場合の方が，データの取得に時間がかかっていることがわかる．これは，外部の API の呼び出しの時間によるものである．

E，F，G は IoT センサのデータと Web サービスのデータを組み合わせたものであり，それぞれ 2 つのセンサデータ，センサデータと Web サービスからのデータ，2 つの Web サービスからのデータである．基盤からのデータの取得を 2 回行うため，応答時間は長くなっている．基盤を用いない場合で API からデータを取得する場合は，IoT 基盤を用いる場合の方が短い応答時間を示している．E の場合，オーバーヘッドは大きくなっているが，約 200ms ほどである．以上の結果から，実装した IoT サービス基盤によって生じるオーバーヘッドは限定的である．また，大規模言語モデルによる回答生成の時間は 4 秒から 5 秒ほどかかるためオーバーヘッドはより限定的である．

次に，大規模言語モデルが表 1 の実装したルールとプロンプトに基づいて生成した回答が想定した回答であるかを検証する．A から G のシナリオについて検証したところ，大規模言語モデルは想定した回答を生成した．例えば，アプリケーションから「現在の研究室の気温」をリクエストすると，「11 月 21 日 23 時 55 分の 208 号室の温度は 18.9 度です．この度数は快適な室温範囲にあるので，特に暖房や冷房を追加する必要はありません．」という回答が得られた．この回答は気温のリクエストに対する回答として適している．

表 1 実装したプロンプトとルール

プロンプト	{$ServiceType:$Value:[$Time]}
ルール	「$Time の$ServiceType は$Value です．」に加えてユーザに有益な情報を 1 文追加.

5. おわりに

本研究では，大規模言語モデルが物理世界の情報に基づく回答を生成するための IoT サービス基盤の提案，実装をした．実装した IoT サービス基盤の応答時間，大規模言語モデルが生成する回答についての評価を行い，提案した IoT 基盤の有用性，実用性を示した．今後アクチュエータを用いたシナリオについて実装する予定である．

謝辞

本研究は，日本学術振興会科学研究費（B）(21H03556，2021 年度～2023 年度)の補助を受けた．

参考文献

[1] S. Aingh Gill, R Kaur. ChatGPT: Vision and challenges. Internet of Things and Cyber Physical Systems, Vol.3, pp.262-271, 2023

[2] A. Al-Fuqaha, et al. Internet of Things: A survey on Enabling Technologies, Protocols and Applications. IEEE Communications Surveys & Tutrials, Vol.17, No.4, pp.2347-2376, 2015

[3]大川楠人ほか，異種の IoT デバイスと Web サービスの相互運用のための IoT サービス基盤，電子情報通信学会論文誌，Vol. J105-D, No.1, pp.41-51, 2022