CO-002

大規模言語モデルを活用した IoT サービス基盤の構築
Realizing IoT Service Infrastructure by Utilizing Large Language Models

横辻　龍太郎†　　林　冬惠†
Ryutaro Yokotsuji　Donghui Lin

1. はじめに

Internet of Things（IoT）は，1999 年に提案されて以降，急速な発展を続けている [1][2]．IoT は，物理世界のモノをインターネットに接続することで，身の回りの情報の取得，モノの遠隔操作を可能にする．IoT センサや IoT アクチュエータの数や種類を増えており，物理世界の様々な情報を取得することが可能になっている．様々な IoT デバイスや Web サービスを組み合わせることで，スマートホームや工場の自動化，車の運転支援などの多くの分野で高度な IoT アプリケーションの実現が可能になっている．しかし，複数のデバイスから取得したセンサデータを有効に活用する IoT アプリケーションの開発は困難である．複数の IoT デバイスを組み合わせて開発する場合，デバイスを開発した企業やデバイスの世代によってメッセージングプロトコル，センサデータのフォーマットなどの仕様が異なっている．また，センサと同種の情報を取得する Web サービスを組み合わせる場合にも同様の問題が発生する．この問題に対処するために，様々な団体によって標準化が行われているが，統一された手法は確立されていない．このようなデータの相互運用性の欠如は，IoT アプリケーション開発における課題になっている [3].

また，IoT センサから取得したデータの活用方法は限られている．例えば，スマートホーム環境では，ユーザからの質問に対して，室内の気温は 20 度です，のように取得したデータの最低限の情報をユーザに与える対話を行う．大規模なテキストデータで訓練された大規模なパラメータを持つニューラルネットワークである大規模言語モデルは，ChatGPT[1] の登場をきっかけに世間の注目を集めている [4]．ChatGPT をはじめとする大規模言語モデルは，ユーザからの質問に対する回答やユーザとの高度な対話を自然言語で行うことができる．IoT デバイスから取得したデータを用いた対話に大規模言語モデルを用いれば，取得したデータの情報に加えて，大規模言語モデルの広範囲の知識を含んだ対話が可能になる．また，近年，大規模言語モデルの活用方法として AI エージェントが注目を浴びている．AI エージェントは，大規模言語モデルを用いることで自律的な行動を高度なコミュニケーション能力を備えた自律的エージェントとしての進化を遂げている [13]．AI エージェントは Web 検索やデータベースの操作をはじめとしたツールの活用をすることで複雑なタスクの遂行も可能であることから様々な分野での活用が期待されている．しかし，大規模言語モデルは高度な対話ができる一方で，身の回りの情報に基づく回答を生成することはできない．大規模言語モデルだけでは，IoT センサから取得した物理世界の情報を利用するための仕組みはないため，物理世界の情報に基づいた回答を生成することはできない．最新の情報やユーザの周囲の情報に基づく回答を生成するためには，大規模言語モデルに必要な情報を与える仕組みが必要である．

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いて回答の生成を行える IoT サービス基盤を構築することである．本研究の目的を達成するために 2 つの課題に取り組む．

・IoT デバイス，Web サービス間での相互運用性の実現

IoT デバイスや Web サービスから取得したデータを活用する事でアプリケーション開発者はより高度な IoT アプリケーションを開発する事ができる．そのため，IoT サービス基盤は IoT デバイスや Web サービスから取得するデータをシームレスに運用できるように設計する必要がある．

・IoT サービスと大規模言語モデルの接続

大規模言語モデルを用いた対話は，過去のテキストデータによって学習されたモデルに基づくものである．物理世界の情報を基にした回答の生成を行うために，IoT サービスと大規模言語モデルの接続が必要である．

これらの課題を解決するために，まず，センサから取得したデータを標準化するインターフェースを設計する．センサや Web サービスから形式の異なるデータを取得する際に，インターフェースを用いて統一された形式に標準化することで，データの異質性を考慮することなくアプリケーションの開発が可能になるようにする．次に，大規模言語モデルが IoT センサや Web サービスのデータを基にした回答を生成できるような仕組みを設計する．具体的には，まず，ユーザからのリクエストに対応した IoT データを大規模言語モデルのプロンプトに変換する．そして，プロンプトに基づいて大規模言語モデルが生成する回答のルールを記述し，大規模言語モデルに与えることで，物理世界の情報に基づいた回答の生成を行えるようにする．

本論文の構成は以下の通りである．2 章では関連研究について述べることで本研究の立ち位置を明確にする．3 章では本研究の提案手法とアーキテクチャについて説明し，それらに基づくアプリケーションの実装について述べる．4 章では，提案手法によるシナリオ評価と実装した IoT サービス基盤の応答時間に関して，IoT サービス基盤を用いない場合と比較して性能を評価する．第 5 章では本論文の結論と今後の展望を述べる．

2. 関連研究

本章では，IoT の相互運用性についての説明と相互運用性を実現するために行われている研究について紹介し，既存の IoT 標準化手法について解説する．そして，大規模言語モデルの IoT 分野での活用の現状および IoT 分野以外での活用事例を紹介し，本研究の立ち位置を明確にする．

2.1 IoT の相互運用性

IoT の発展を妨げる大きな要因の 1 つとして，相互運用性の問題が挙げられている．相互運用性の問題は複数の要

†岡山大学　Okayama University
1 https://chat.openai.com/

Copyright © 2024 by
The Institute of Electronics, Information and Communication Engineers and
Information Processing Society of Japan All rights reserved.

素から生じている [7]．1 つ目は，デバイスの通信プロトコルによるものである．デバイスの通信プロトコルは開発した企業，デバイスの世代などによって異なっている．そのため，標準化されたプロトコルが必要になるので，MQTT プロトコルが広く使われている [8]．2 つ目は，受け取ったデータのセマンティック相互運用性と，データ形式の相互運用性である．これらは主に，デバイスから受け取ったデータに関する相互運用性である．データの形式は JSON，XML，CSV など様々な形で記述されるが，一貫したデータ形式で記述して他の形式を扱う際に適切に変換する構造が必要になる．3 つ目は，IoT プラットフォーム間での相互運用性である．これは，1 つのアプリケーションに対して複数の IoT プラットフォームを使用する際に生じる問題で，プラットフォーム間でのシームレスな連携を行うことを想定したプラットフォームの構築が必要になる [7]．一方で，標準化のための取り組みとして，標準化団体による IoT のためプロトコルやアーキテクチャが開発されている．Web の標準化団体である W3C によって Web の標準化技術を使って提案された Web of Things（WoT）は，モノがどのように構成されているかを記述した Things Description を用いて相互運用を行なっている [5]．しかし，WoT は IoT デバイスのデータのドメイン固有の語彙が存在しない [9]．Novo らはそのような問題を解決するために，具体的な利用シナリオを与えている [9]．大川らは，対話エージェント環境で IoT サービスを標準化するために IoT デバイスの情報にメタデータを与え，センサやアクチュエータの種類ごとの標準化インターフェースを実装することで IoT の相互運用を実現している [6].

本研究では，大川らの標準化手法を基に利用シナリオを想定したインターフェースによる標準化を行う [6]．ただし，既存研究では実装した IoT サービス基盤をユーザの状況を考慮した対話エージェントの実現のために活用しているが，本研究では，大規模言語モデルに活用するという点で異なっている．また，大川らの実装した IoT サービス基盤では，イベントのルールを専用のツールの複雑なプログラムによって 1 つ 1 つのイベントの処理を記述しているが，本研究ではルールは大規模言語モデルに与えるため，自然言語で記述することができるので，複雑なプログラムを用いる必要はない．さらに，1 つのルールでセンサを利用したシナリオに対応できるので，ルールの数も削減される．

2.2 大規模言語モデルの IoT への活用

近年，大規模言語モデルがテーマの研究や，大規模言語モデルを活用した研究は盛んに行われるようになってきている．特に，大規模言語モデルベースの AI エージェントが注目を浴びている [16]．例えば，25 の AI エージェントに人間的で協調的な振る舞いを行わせることで小規模な社会のシミュレーションを行う研究や，ゲーム内のプレイヤーの周囲の情報から大規模言語モデルが必要な行動を推論してタスクを行うためのコードを生成する研究が注目を浴びている [10][14]．また，医療，金融，教育，交通などの複雑なタスクに適応可能な AI エージェントのフレームワークに関する研究が行われている [11]．化学の分野では，専門家が設計したタスク実行のためのツールを用いることで専門的な知識を要するタスクの実行を行える化学エージェントが開発されている [15]．一方で，IoT 分野での大規模言語モデルの活用の事例は多く見られない．ユーザによる自然言語のリクエストから場所や IoT サービスの種類を含むパラメータを取得して，マイコンの制御を行なって対話を行う研究が行われており，自然言語処理のための活用事例が存在する [12]．本研究は IoT 分野で AI エージェントを適用するための初期的な研究である．そのために，IoT データから取得したデータをプロンプトとして与え，プロンプトからユーザに表示する回答や，アクチュエータを制御するための回答生成に大規模言語モデルを活用する．

本研究では，実世界での利用シナリオ想定したルールとセンサデータに基づくプロンプトを定義する．それらを与えることで大規模言語モデルが物理世界の情報に基づいて回答を生成することを可能にする．

3. IoT サービス基盤の構築

本章では，まず，異なるデータ形式を持つ IoT データを相互運用するための標準化手法を述べる．次に，大規模言語モデルが物理世界の情報に基づいて回答を生成するための仕組みを説明する．そして，2 つの手法を基に設計したアーキテクチャを解説し，アーキテクチャを基に設計したアプリケーションの説明を行う．

3.1 IoT サービスの相互運用性の実現

本研究では，IoT サービスの相互運用性を実現するために，データの標準化を行う．そのために，階層化されたインターフェースを実装する．異種デバイスや Web サービスの異質性を吸収する標準インターフェースの構造について説明する．センサデータの基本的なパラメータを持つ Sensor 抽象クラスを定義して，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，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，データを活用する際に IoT サービスごとの記述をする必要がなくなるため，データを相互運用することができる．

具体的な標準化の例を図 1 に示す．気温を取得する Web サービスである OpenWeather[1] から取得したデータは JSON 形式で記述されているが，気温を取得する IoT センサデバイスである DHT11 のデータは，{15.4, 20231121-22:02:38} のような形式である．どちらも気温を取得する IoT サービスなので，TemperatureSensor クラスを継承したそれぞれのアダプタには getTemperature メソッドがある．メソッド内にそれぞれのデータの形式の違いを吸収する記述を行うことで，getTemperature メソッドによる統一された温度データの取得が可能になる．

1 https://openweathermap.org

Copyright © 2024 by
The Institute of Electronics, Information and Communication Engineers and
Information Processing Society of Japan All rights reserved.

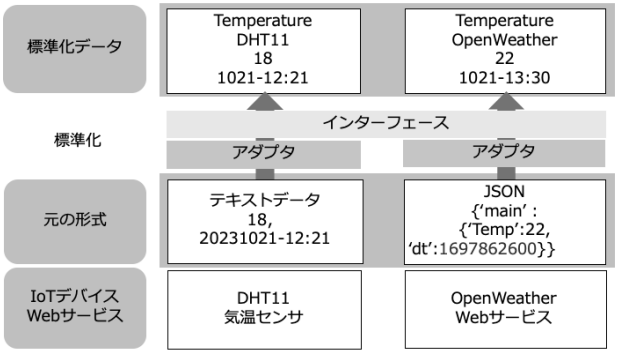

図 1　標準化の例

3.2 IoT サービス基盤と大規模言語モデルの接続

大規模言語モデルは，ユーザーからの自然言語による質問のリクエストをプロンプトとして受け取り，自然言語による回答を生成する．大規模言語モデルに回答のルールを与えることで，回答を生成する際に，ルールに基づいた回答を行うことができる．本研究では，取得した IoT データを活用するために大規模言語モデルが生成する回答の指示のためのプロンプトを定義することで，大規模言語モデルが物理世界の情報に基づいて回答することを可能にする．また，標準化されたデータをルールの記述に基づいた形式に変換してプロンプトとして扱う．

表 1 に実装したサービスの種類とサービスに対応したセンサ情報に基づくプロンプトを示す．センサに基づく情報推薦とはセンサからの情報に基づいて大規模言語モデルがユーザに情報を推薦することを想定したサービスである．アクチュエータ制御は，センサから取得した情報からアクチュエータを制御するサービスである．

表 1　サービスとセンサ情報に基づくプロンプト

サービス	プロンプト
単一センサに基づく情報推薦	[$ServiceType:$Value:[$Time]:$Place]
複数センサに基づく情報推薦	[$ServiceType1:$Value1:[$Time]:$Place1]#[$ServiceType2:$Value2:[$Time]:$Place2]
アクチュエータ制御	[$ServiceType:$Value:[$Time]:$Place]

次に，大規模言語モデルに与えるセンサ情報に基づくプロンプトについて説明する．IoT サービス基盤では，リクエストに応じて必要なデータをプロンプトの一部に変換する．$ServiceType は IoT サービスの種類を表したもので，$Value，$Time，$Place は，それぞれデータの値と時間と場所を示したものである．2 つのデータを扱うときは，それぞれのデータを 1 つのプロンプトに変換してから連結することで 1 つのプロンプトとして扱う．

次に，ルールの役割を担う回答生成のためのプロンプトについて説明する．定義した内容は表 2 に示す．大規模言語モデルにリクエストする際にセンサデータに基づくプロンプトに加えて対応した回答生成のためのルールのプロンプトを与えることで大規模言語モデルはプロンプトから想定した回答を生成することができる．また，従来の IoT アプリケーションでは，プログラムによって記述された厳密なルールが必要であるが，提案手法では，自然言語で記述して大規模言語モデルに与えることができるので，より柔軟なサービスの実装が可能になる．

表 2　回答生成のためのプロンプト

プロンプトに対して「$Time の$Place の$ServiceType は $Value です．」に加えて，$ServiceType と$Value を考慮したユーザに有益な情報の 1 文.
プロンプトに対して「$Time の$Place1 の$ServiceType1 は $Value1 で$Place2 の$SrviceType2 は$Value2 です．」に加えて，$ServiceType1，$ServiceType2 と$Value1，$Value2 を考慮したユーザに有益な 1 文.
$ServiceType と$Value を考慮して対応したアクチュエータ $Actuator と状態 $Value1 を決定して，{“Actuator” : $Actuator, “Place” : $Place, “Value” : $Value1}の形式で回答を生成.

3.3 基盤のアーキテクチャ

ここでは，3.1 章と 3.2 章で述べた手法を用いて IoT サービス基盤の構築について説明する．IoT サービス基盤は，IoT デバイスとアプリケーションを繋ぐ中間に位置する．主な役割は，IoT デバイスや Web サービスとのデータのやり取り，収集したデータの分析，アプリケーションからの要求への返答である．図 2 に，IoT サービス基盤を含めた全体のアーキテクチャを示す．

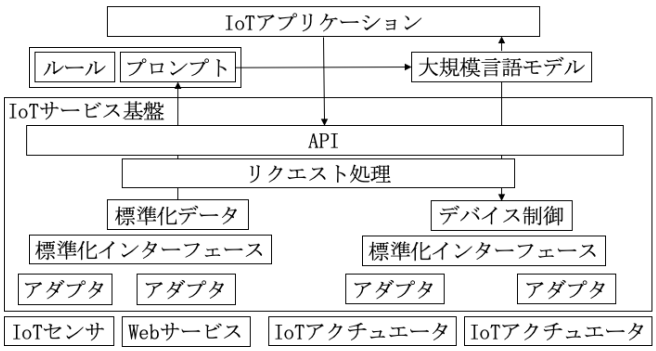

図 2　提案するアーキテクチャ

IoT センサ，Web サービスはどちらも情報を取得するためのものであり，IoT センサは身の回りから，Web サービスは Web から取得して IoT サービス基盤に情報を送信する．一方で，IoT アクチュエータは，IoT サービス基盤からの制御で駆動する．IoT サービス基盤に送信されたデータやアクチュエータに送信するためのデータを標準化するためにアダプタおよび標準化インターフェースを用いる．各デバイスに対応したアダプタによってデータの違いを吸収する．また，IoT サービス基盤は，アプリケーションからのリクエストを API を介して受け取ることでリクエスト処理を行う．大規模言語モデルが回答を生成する際には，IoT サービス基盤からのデータをプロンプトに変換し，ルールとともに大規模言語モデルに与える．

Copyright © 2024 by
The Institute of Electronics, Information and Communication Engineers and
Information Processing Society of Japan All rights reserved.

3.4 実装

本研究では，提案手法に則って実世界での利用を想定したアプリケーションを実装する．具体的には，スマートホーム環境での利用を想定したユーザへの情報推薦のアプリケーションを実装する．アプリケーションには，ユーザのリクエストに応じてセンサ情報に基づく対話を表示するサービスとセンサ情報に基づいて定期的にアクチュエータを制御するサービスを実装する．実装の詳細は，図 3 に示す．

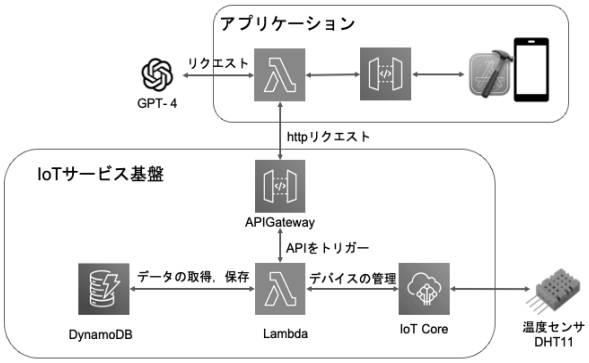

図 3　実装図

本研究の実装の大部分は Amazon Web Services[1]（AWS）を用いて実装する．IoT センサは IoT サービス基盤上の IoT Core に接続されてデータの通信を行う．基盤で受け取ったデータを AWS Lambda 上のインターフェースを用いて標準化して DynamoDB にデータを蓄積する．IoT サービス基盤とアプリケーションは API Gateway の API を通じて通信を行う．アプリケーションは XCode を用いて iOS アプリケーションとして実装し，API を介してリクエストを送る．IoT サービス基盤から受け取ったプロンプトとルールから大規模言語モデルは回答の生成を行い，アプリケーションは回答を表示する．

4. 評価

本章では，実世界での利用を想定したシナリオに基づいて実装した IoT サービス基盤の性能および大規模言語モデルが生成する回答に関する評価を行う．先行研究 [6]では，シナリオ評価と応答時間の評価に加えて IoT サービスの標準化によるサービスのバリエーション数の評価がされており，バリエーション数が削減できることが示されていたので，本論文ではシナリオ評価と応答時間の評価を行う．

4.1 シナリオ評価

表 3 に実装したシナリオを示す．提案手法によって大規模言語モデルが生成した回答が想定した回答であるかどうかについて検証する．

表 4 に用意したシナリオに対して生成した回答を示す．表中の下線が引かれている箇所は，プロンプトから与えられたパラメータであり，引かれていない箇所は大規模言語モデルによって生成された文章である．A から D までのシナリオは，1 つのデータからなるプロンプトを用いて大規模言語モデルに回答をリクエストしたものである．定義したルールに則って，それぞれのデータに関する 1 文とユーザに有益な情報を 1 文生成していることがわかる．例えば，シナリオ A の場合，「11 月 21 日 23 時 55 分の 208 号室の気温は 18.9 度です．」という 1 文はプロンプトに基づく 1 文であり，「この温度では，薄手のジャケットやカーディガンが適しています．」という 1 文は，大規模言語モデルがプロンプトの情報を考慮してルールに従って生成した 1 文であるため，想定し得る回答である．

表 3 実装したシナリオ

ID	シナリオ	IoT サービス 1	IoT サービス 2
A	室内の温度	DHT11	
B	室外の温度	OpenWeather	
C	室内の湿度	DHT11	
D	室外の湿度	OpenWeather	
E	室内の不快指数	DHT11	DHT11
F	室外の不快指数	OpenWeather	OpenWeather
G	室内外の温度差	DHT11	OpenWeather
H	ライトの点灯	仮想照度センサ	仮想 LED

表 4　生成された回答

ID	回答
A	11 月 21 日 23 時 55 分の 208 号室の温度は 18.9 度です．この温度では，薄手のジャケットやカーディガンが適しています．
B	11 月 21 日 23 時 19 分の岡山市北区の温度は 9.79 度です．寒い夜になりますので，暖かい服装をして外出すると良いでしょう．
C	11 月 21 日 23 時 55 分の 208 号室の湿度は，51%です．湿度が 50%前後であれば，屋内環境は快適とされています．
D	11 月 21 日 23 時 19 分の岡山市北区の湿度は 70.0%です．このような高湿度は霧が発生する可能性があるため，運転時には視界確保に注意してください．
E	11 月 21 日 23 時 55 分の 208 号室の温度は，18.9 度で，湿度は 51.0%です．不快指数は 63.8 で，比較的快適な室内環境と言えます．
F	11 月 21 日 23 時 19 分の岡山市北区の温度は 9.79 度で湿度は 70.0%です．加えて，不快指数は 51 です．この数値ではかなり肌寒く感じるため，防寒対策をしっかりと行うことをお勧めします．
G	11 月 21 日 23 時 19 分の 208 号室の温度は 18.9 度で，岡山市北区の温度は 9.79 度です．温度差は，9.11 度です．夜間に室内と屋外の温度差が大きいため，暖房設備の利用を考慮すると快適に過ごせるでしょう．
H	{“Acutuator”: “Light”, “Place”: “208”, “Value”: “On”}

シナリオ E から G は，2 つのデータからなるプロンプトに基づいて回答をリクエストしている．全てのシナリオの回答が，2 つのデータの情報を示す 1 文に加えて，2 つのデータを考慮した文章とその情報に関して有益な情報からなる 1 文を生成している．例えば，シナリオ G は，IoT セ

1 https://aws.amazon.com/

Copyright © 2024 by
The Institute of Electronics, Information and Communication Engineers and
Information Processing Society of Japan All rights reserved.

ンサからの温度データと Web サービスからの温度データを利用したシナリオで，2 つの温度情報に関する 1 文と 2 つの気温の差とその値に関する有益な情報を含む文章を生成することを想定している.実際の生成結果は，想定した対話内容を生成していることがわかる．

次に，センサの情報に基づいてユーザに情報を表示するシナリオとは異なる，アクチュエータを利用したシナリオについての評価を行う．アクチュエータ制御のシナリオでも同様に，センサデータをプロンプトに変換し大規模言語モデルによる回答生成を行う．その後，生成された回答は，IoT サービス基盤に送信されてアクチュエータを制御する．シナリオ H の回答は，仮想の照度センサから照度が低いデータを与えた場合の生成結果である．生成された回答は，208 号室のライトをオンにするというプログラムが処理可能な形式の回答であり，IoT サービス基盤に送信されて仮想の LED がオンになっていることが確認できた．

以上の結果から，センサを利用したシナリオに加えて，アクチュエータを利用したシナリオも実現できているので，大規模言語モデルが物理世界の情報に基づいて回答を生成できていることがわかる．

4.2 応答時間の評価

実装したシナリオが IoT サービス基盤を使う場合と，IoT サービス基盤を通さないで直接データを取得する場合の応答時間を比較することで，IoT サービス基盤のオーバーヘッドを検証する．計測したのは，ユーザがアプリケーションにリクエストしてからアプリケーションに結果が表示されるまでの時間である．IoT サービス基盤を介さない場合，IoT センサデバイスに関しては，DynamoDB に保管したデータを直接取得し，Web サービスに関しては，API により直接データを取得する．実験は，XCode のシミュレータ機能によって，仮想の iPhone 上で行う．iPhone 上にインストールされた実装したアプリケーションからリクエストを各シナリオのリクエストのためのボタンをタップしてから結果が表示されるまでの時間を測定する．

最初に，大規模言語モデルによる回答生成の時間を除いた応答時間についての評価を行う．図 4 は，実装したシナリオについて回答生成の時間を除いた 20 回の応答時間の平均である．まず，1 つの IoT サービスを利用したシナリオの応答時間の評価を行う．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と考えられる．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 サービス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 サービス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．これは，外部の API の呼び出しの時間によるものである．続いて，2 つの IoT サービスを利用したシナリオの応答時間の評価を行う．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 サービス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 サービス基盤を用いることによって生じるオーバーヘッドは限定的であることがわかる．

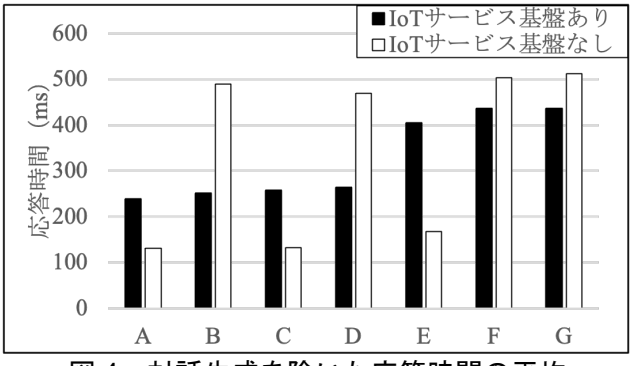

図 4　対話生成を除いた応答時間の平均

次に，大規模言語モデルによる回答生成を含めた応答時間の評価を行う．大規模言語モデルによる回答の生成の時間は数秒程度かかると予想されるため，IoT サービス基盤によって生じるオーバーヘッドが応答時間に占める割合は小さいものになると考えられる．実際に 20 回計測した平均の応答時間を図 5 に示す．まず，1 つの IoT サービスを利用したシナリオの応答時間の評価を行う．1 つの IoT センサからなるシナリオ A と C は，回答生成を含めない応答時間と同様に IoT サービス基盤を用いない方がより短時間での応答を示しているが，オーバーヘッドが増加していることがわかる．1 つの Web サービスを利用したシナリオ B，D の応答時間は，対話生成を除いた応答時間はどちらもデータを直接取得する方が長い時間がかかっていたが，B の全体の応答時間の差は限りなく 0ms である．また，D の場合は対話生成を除いた時間と同等の差が生じている．続いて，2 つの IoT サービスを利用したシナリオの応答時間の評価を行う．2 種類のセンサデータを利用したシナリオ E は，対話の生成時間を除いた応答時間と比較して応答時間の大小関係は変化していないが，200ms 程度であったオーバーヘッドが 1,000ms 程度まで増加している．シナリオ F は対話生成にかかる時間を加えても応答時間の差は 200ms 程度で大きな変化は見られない．シナリオ G は，元々のオーバーヘッドは 600ms ほどであるが，対話生成の時間を加えると 1,000ms ほどまで増加している．多くのシナリオに

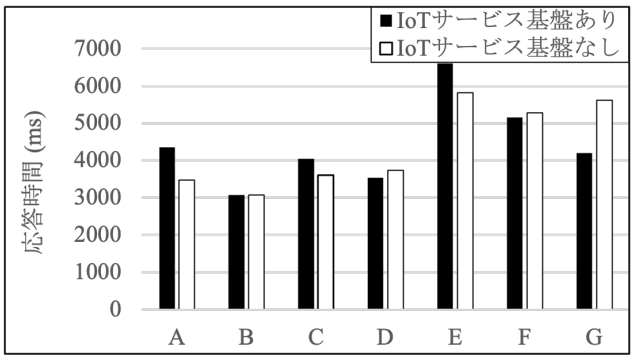

図 5　全体の応答時間の平均

Copyright © 2024 by
The Institute of Electronics, Information and Communication Engineers and
Information Processing Society of Japan All rights reserved.

関して，対話生成の時間を含むことでオーバーヘッドの増減が起こっている．この原因は大規模言語モデルによる回答生成にかかる時間が一定でないことにある.リクエスト毎に生成時間が平均して約 4 秒かかることや数秒程度異なることから，データ取得の際に 100ms や 200ms 程度のオーバーヘッドが存在したとしても，全体の応答時間に占める割合は小さいものであると考えられる．

4.3 考察

本章では，実装した IoT サービス基盤によるオーバーヘッドが限定的であることと，実世界の利用シナリオに基づいて，大規模言語モデルによって生成された対話内容が適切であることを示した．応答時間に関して，生成時間を短縮するためにルールの記述をより簡潔にすることで対話生成にかかる時間が短縮すると考えられる．大規模言語モデルが生成する回答は常に決まったものではなく，生成の度に異なる回答を生成する可能性がある．大規模言語モデルが一貫した回答を生成できるようにするためのプロンプトの定義や仕組みが必要になる．加えて，本研究の提案手法がより複雑なシナリオやスマートホーム以外の産業，交通，医療の IoT サービスが活用な可能な分野での適用が可能か検証を行う必要がある．

今回は，大規模言語モデルを活用して IoT アプリケーションを実装したが，今後は，AI エージェント環境で IoT アプリケーションの実装を行う．また，本研究では，大規模言語モデルに与えるプロンプトは事前に定義した．一方で，AI エージェントを実装する際には，ユーザの入力から柔軟に対応できるプロンプトを自動で設計する仕組みが必要になる．加えて，大規模言語モデルがタスク実行に必要な IoT サービスを選択して取得した情報から回答を生成，アクチュエータの制御を行うことができるプロンプトを大規模言語モデルが生成できるような仕組みが必要になる．今後は，そのような仕組みを実現するための研究を行なっていく．

5. おわりに

本論文では，大規模言語モデルを活用した IoT サービス基盤の構築を行なった．大規模言語モデルが物理世界の情報に基づいた回答の生成を可能にするための IoT サービス基盤を構築するために IoT サービスの相互運用性の実現と IoT と大規模言語モデルの接続という 2 つの課題に取り組んだ．まず，IoT サービスの標準化を行うためのインターフェースを設計した．インターフェースによって IoT サービスを利用する際に，デバイスの異質性を考慮しないで実装を行えるようにした．次に，IoT と大規模言語モデルの接続のために，取得した IoT データを大規模言語モデルのプロンプトに変換し，実世界での利用シナリオに応じたルールを定義することで，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ようにした．

提案手法を基に，スマートホーム環境を想定したアプリケーションを実装し，シナリオ評価と応答時間の評価を行った．シナリオ評価から大規模言語モデルがセンサから取得した IoT データに基づいて，想定した回答の生成が適切に行えているかを確認した．また，センサデータから大規模言語モデルが判断して，アクチュエータを制御するための回答を生成するシナリオについても検証し，実現が可能であることを示した．そして，IoT サービス基盤を用いることによって発生するオーバーヘッドを検証した．実験の結果，オーバーヘッドが全体の応答時間に占める割合に対して，大規模言語モデルによる対話生成の時間が全体の応答時間に占める割合の方がはるかに大きいことからオーバーヘッドの影響は限定的であると考えられる．今後は，IoT サービスを用いた AI エージェントを構築する予定である．

謝辞

本研究は，日本学術振興会科学研究費（B）(21H03561, 24K03001)の補助を受けた．

参考文献

[1] Kevin Ashton. That ‘Internet of Things’ thing. RFID Journal, 22(7):97-114, 2009.
[2] Ram D Sriram and Amit Sheth. Internet of Things perspectives. IT Professional, 17(3):60-63, 2015.
[3] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The Internet of Things: A survey. Computer Networks, 54(15):2787-2805, 2010.
[4] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Zican Dong, et al. A survey of Large Language Models. arXiv preprint arXiv:2303.18223, 2023.
[5] W3C. W3c Web of Things. https://w3c.org/WoT/, 2017.
[6] 大川楠人, 林冬惠，村上陽平，中口孝雄．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 D，105(1):41-51, 2022.
[7] Mahda Noura, Mohammed Atiquzzaman, and Martin Gaedke. Interoperability in Internet of Things: Taxonomies and open challenges. Mobile Networks and Applications, 24:796-809, 2019.
[8] Dipa Soni and Ashwin Makwana. A survey on MQTT: a protocol of Internet of Things (IoT). International Conference on Telecommunication, Power Analysis and Computing Techniques (ICTPACT-2017), 20:173-177, 2017.
[9] Ocar Novo and Mario Di Francesco. Semantic interoperability in the IoT: extending the Web of Things architecture. ACM Transactions on Internet of Things, 1(1):1-25, 2020.
[10] Joon Sung Park, joseph C. O’Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, 2:1-22, 2023
[11] Yashar Talebirad and Amirhossein Nadiri. Multi-agent collaboration: Harnessing the power of intelligent LLM agents. arXiv:2306.03314, 2023.
[12] George Alexakis, Spyros Panagiotakis, Alexander Fragkakis, Evangelos Markakis, and Kostas Vassilakis. Control of smart home operations using natural language processing, voice recognition and IoT technologies in a multi-tier architecture. Designs, 3(3):32, 2019.
[13] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, et al. The rise and potential of Large Language Model based agents: A survey. arXiv:2309.07864, 2023.
[14] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, et al. Voyager: An open-ended embodied agent with Large Language Models. arXiv:2305.16291, 2023.
[15] Andres M Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, Philippe Schwaller. ChemCrow: Augmenting Large-Language Models with chemistry tools. arXiv:2304.05376, 2023.
[16] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, et al. A survey on Large Language Model based autonomous agents. Frontiers of Computer Science, 18(6): 1-26, 2024.

Copyright © 2024 by
The Institute of Electronics, Information and Communication Engineers and
Information Processing Society of Japan All rights reserved.