LLMによる意図解釈に基づく Verifiable Credentials提示手法

藤井 孝輔[1,a)]　渡邉 健[1]　大塚 巧巳[1]　佐古 和恵[1]

概要：Verifiable Credentials (VC) は、暗号技術を用いて人や物の情報を電子的に保証する仕組みである。その特徴的な機能の一つに、VC の中から特定の属性のみを選んで提示できる選択的開示がある。これにより、プライバシーを保護しつつ、必要な情報だけを証明することが可能となる。しかし、学歴、職歴、資格証明など、個人の活動が多様化するにつれて保有する VC の数も増加する。その結果、例えば就職活動のような特定の目的のために、多数の VC の中から最適な証明書や属性の組み合わせを手動で探し出し、選択する作業は保有者にとって大きな負担となりうる。本研究ではこの課題に対し、保有者が自然言語で意図を指示するだけで、指示に合致する VC と関連する属性を自動で抽出し、提示用のデータを生成するシステムを提案する。システムの中核には、予め特定のタスク向けにファインチューニングしたものをインストールすることにより、ユーザーのローカル環境で独立して動作するオープンソースの大規模言語モデル（LLM）を利用する。これにより、VC に含まれる情報を外部サーバに送信することなく、手元環境でのみ動作することによって、プライバシーを尊重したシステムを実現できる。本論文では、提案システムの全体アーキテクチャ、複数のオープンソースの LLM モデルで実施したファインチューニングの精度比較評価や、学習に用いた独自データセットの構成について述べる。

A Method for Presenting Verifiable Credentials by Interpreting Holder's Intent using LLM

Kosuke Fujii[1,a)]　Ken Watanabe[1]　Takumi Otsuka[1]　Kazue Sako[1]

Abstract: Verifiable Credentials (VCs) provide a cryptographic framework for electronically attesting to information about individuals or objects. A key feature is selective disclosure, which allows a holder to present only specific attributes from a credential, thereby enabling proof of necessary information while preserving privacy. However, as individuals accumulate a diverse range of VCs for academic achievements, professional history, and certifications, the process of manually locating and selecting the optimal combination for a specific purpose, such as a job application, becomes a significant burden. To address this challenge, this paper proposes a system that automates the extraction of relevant VCs and their attributes. By interpreting a user's intent from natural language instructions, the system identifies the appropriate credentials and generates the required data for presentation. The core of this system is an open-source Large Language Model (LLM), specifically fine-tuned for this task, which operates independently within the user's local environment. This on-device architecture is crucial as it ensures that sensitive information within the VCs is never transmitted to an external server, thus offering a solution that is both effective and privacy-preserving by design. This paper details the proposed system's overall architecture, presents a comparative evaluation of fine-tuning accuracy across multiple open-source models, and describes the composition of the custom dataset developed for their training.

[1] 早稲田大学
Waseda University
[a)] kou66@fuji.waseda.jp

1. はじめに

近年、デジタル社会の進展とともに、オンラインでの本人確認や資格証明の重要性が高まっている。その中で、暗号技術を用いて個人や組織の情報を電子的に保証する Verifiable Credentials（VC）[1] が注目を集めている。VC の持つ選択的開示機能は、証明書の中から必要な属性情報のみを開示できるため、プライバシーを保護しつつ、必要な情報だけを相手に提示することを可能にする。

現在、VC の主な利用シナリオは、検証者（Verifier）からの要求に保有者（Holder）が応じる形で提示する、という受動的なものが想定されている。しかし、VC の活用方法はそれだけにとどまらず、様々な可能性が考えられる。例えば、就職活動で自身のスキルを証明するために、語学能力や IT スキルに関する証明書を提示したり、専門家として SNS で発信する際に、その内容が自身の専門分野に関するものであることを示すために所属組織の在籍証明を添付したりするなど、保有者が自身の意図に基づき、能動的に提示内容を選択する場面も考えられる。

今後、社会活動のデジタル化が進み、個人の学歴、職歴、会員資格、研修修了証明といった様々な情報が VC として発行されるようになれば、個人が保有する VC の数は増加していくと考えられる。その結果、保有する VC が数十、数百と増えていった場合、たとえ自発的な情報開示であっても、その中から目的に合った最適な証明書や属性の組み合わせを探し出す作業は負担となりうる。どの VC にどの情報が含まれているかを正確に把握し、過不足なく情報をまとめる作業は、時間と手間を要するだけでなく、誤りの原因ともなりかねない。

本研究ではこの課題に対し、保有者が自然言語で自身の意図を指示するだけで、要求に合致する VC と関連する属性を自動で抽出し、提示用のデータを生成するシステムを提案する。本システムの中核には、特定のタスク向けにファインチューニングされ、ユーザーのローカル環境で独立して動作するオープンソースの大規模言語モデル（LLM）を利用する。これにより、VC に含まれる機微な個人情報を外部のサーバに送信することなく、手元の環境でプライバシーを保護しながら、VC 活用の利便性を向上させることを目指す。

本論文では、提案システムの全体アーキテクチャを述べ、学習に用いた独自データセットの構成と、複数のオープンソースの LLM で実施したファインチューニングの精度比較評価について記述する。

2. システムの構成

本研究では、ユーザーが自然言語で自身の意図を指示するだけで、要求に合致する VC と関連属性を自動で抽出し、VC を提示するためのデータフォーマットである Verifiable Presentation (VP) を生成するシステムを設計・実装した。本章では、提案システムの全体アーキテクチャとその構成要素、およびデータ処理のフローについて、図 1 に沿って述べる。

2.1 アーキテクチャと処理フロー

提案システムは、図 1 に示すように、大きく分けて「VC フィルタリング」「DCQL 生成」「VP 生成」の 3 つの主要なコンポーネントで構成される。ユーザーからの自然言語指示を起点とし、これらのコンポーネントが順次処理を行うことで、最終的に VP が生成される。

2.1.1 ステップ 1：VC フィルタリング

処理の最初のステップは、ユーザーからの「`I want to prove my English skills.`」といった自然言語の指示を受け、ユーザーのウォレットアプリ内に保管されている多種多様な VC の中から、関連性の高いものを少数絞り込むことである。このフィルタリングにより、後続の LLM に、ユーザーの指示に無関係な VC 情報を入力することによる出力精度の低下や、応答遅延を防ぐことが期待できる。本ステップの具体的な実装については、第 3 章で詳述する。

2.1.2 ステップ 2：DCQL の生成

次に、フィルタリングされた VC とユーザーの指示内容を基に、LLM を用いて VP を要求するためのクエリを生成する。本研究では、このクエリ言語として **Digital Credentials Query Language** (DCQL) [2] を採用する。DCQL は、OpenID for Verifiable Presentations 1.0 [2] で定義されているクエリ言語であり、検証者が VC の種類や形式、さらには特定の属性を選択的に指定して提示を要求するために用いられる。このクエリ生成プロセスの詳細については、第 4 章で述べる。

2.1.3 ステップ 3：VP の生成

最終ステップでは、生成された DCQL クエリを実行し、VP を生成する。DCQL クエリは、どの VC からどの属性を抽出するかを指示しており、この指示に従って VC から VP が生成される。そして、W3C の Verifiable Credentials Data Model[1] に準拠した形式で VP が組み立てられ、ユーザーに提示される。

本システムは、これら一連の処理をユーザーのローカル環境で完結させる。その中核となる LLM には、「ユーザーの自然言語指示と VC をもとに適切な DCQL を生成する」タスクに特化してファインチューニングしたオープンソースのモデルを利用する。これにより、VC に含まれる機微な情報を外部のサーバに送信することなく、プライバシーを保護しながら安全に VP を生成することが可能となる。

2.2 対象とする Verifiable Credentials

本システムが対象とする VC は、W3C が標準化を進め

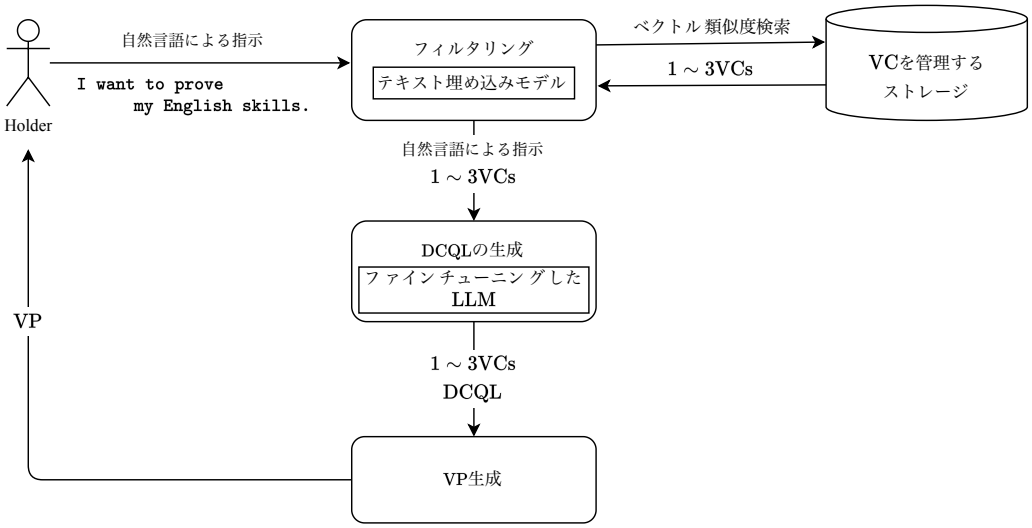

図 1 システムの構成

る Verifiable Credentials Data Model v2.0（VCDM）[1] に準拠するものとする。VCDM は、JSON-LD フォーマットに基づいたデータモデルであり、その構造はいくつかの主要なプロパティによって定義される。具体的には、VC を識別するための `id`、VC の種類を示す `type`、発行者情報を示す `issuer`、有効期間を定義する `validFrom` や `validUntil` などが挙げられる。`credentialSubject` には、保有者に関する具体的な属性情報（クレーム）が記述される。図 2 に、具体的な構造の例を示す。本研究のシステムは、LLM を用いてユーザーの自然言語指示を解釈し、この `credentialSubject` 内に記述された具体的な属性情報などを抽出する DCQL クエリを生成する。

```
{
  "@context": ["https://www.w3.org/ns/credentials/v2"],
  "id": "http://example.com/credentials/10",
  "type": ["VerifiableCredential",
           "EmployeeIDCredential"],
  "issuer": {
    "id": "https://techcorp.jp",
    "name": "Tech Corporation"
  },
  "validFrom": "2020-01-01T00:00:00Z",
  "credentialSubject": {
    "employeeId": "EMP-2024-001",
    "employeeName": "Taro Yamada",
    "department": "Technology Development",
    "position": "Senior Engineer",
    "joinDate": "2020-04-01",
    "employmentType": "Full-time"
  }
}
```

図 2 VCDM の構造例

2.3 研究のスコープ

本研究は、ユーザーの自然言語の指示に基づき、単一の VC から VP を生成することに焦点を当てる。そのため、複数 VC から属性を統合して VP を生成する設計および実装は行わない。また、データフォーマットとしては、VCDM のみを対象とし、SD-JWT [3] や mDL [4] など他のフォーマットは扱わない。VC へのクエリとしては DCQL のみを対象とし、Presentation Exchange [5] や SPARQL [6] のような他のクエリ言語との比較は今後の課題とする。

3. フィルタリング

本研究で提案するシステムは、LLM を用いてユーザーの自然言語指示を解釈し、ユーザーが保有する多数の VC の中から、提示すべき VC の情報を正確に抽出することを目的とする。しかし、LLM が一度に処理できる情報量には、コンテキストウィンドウと呼ばれる制限が存在する。この制限を超過する入力は、モデルに受け付けられないか、あるいは推論性能の著しい低下を招くことが知られている。

ユーザーが保有する VC が大量である場合、それら全ての情報を LLM のコンテキストウィンドウ内に収めることは困難となる。仮に収めることができたとしても、ユーザーの指示と無関係な VC の情報を大量に入力することは、最終的な出力精度を低下させる要因となり得る。

そこで本システムでは、LLM による処理の前段で、多数の VC の中からユーザーの指示と関連性の高いものを少数に絞り込むフィルタリング機構を導入する。この事前処理により、LLM は関連性の高い情報のみに集中でき、計算資源を効率的に利用しつつ、高精度な処理を実現することが期待される。

ただし、このフィルタリング処理で候補を過度に絞り込みすぎると、本来提示すべきだった VC を誤って排除してしまうリスクが生じる。そこで本システムのフィルタリングでは、最終的な VC を 1 つに特定するのではなく、ユーザーの指示に少しでも関連する可能性のある VC を、あえ

て複数、網羅的に候補として選出する設計とする。後続のLLM は、このようにして絞り込まれた候補群の中から、指示の文脈や意図をより深く解釈し、最終的に提示すべきVC を厳密に選び出す役割も担う。

本章では、このフィルタリングの具体的な設計と実装、およびその性能評価について詳述する。

3.1 フィルタリングの設計と実装

本フィルタリング機構は、ユーザーの自然言語指示とVC 群との意味的な関連性を捉えるため、テキスト埋め込み（Text Embedding）技術を利用する。これにより、VC 群と自然言語による指示を共通のベクトル空間上で比較し、その類似度を定量的に評価することが可能となる。ここで関連性が高いと思われた VC をフィルタリングする。

処理フローは以下の 3 ステップで構成される。

(1) **テキスト埋め込みのための前処理**:
テキスト埋め込みの精度と応答速度は、入力文字列の質に影響される。本研究では、VC のような JSON データを入力として直接ベクトル化を行うのではなく、テキスト埋め込みへの入力となる文字列を生成する前処理を実行する。この処理では、VC から不要な情報や識別性の低い情報を削ぎ落とし、その本質的な特徴を簡潔に表現した文字列を生成する。具体的には、VC を特徴づける情報として、`type`（VC の種別）、`issuer`（発行者名）、および `credentialSubject`（属性情報）内の全フィールドを対象とする。これらの情報を連結し、意味的に重要である一つの文字列を生成する。この際、識別性の低い情報、例えば全ての VC に共通する`"VerifiableCredential"`というタイプ名や、有効期限などのフィールドは、テキスト埋め込みへの入力文字列から削除される。

(2) **テキスト埋め込みによる数値ベクトル化**:
前処理によって生成された各 VC の文字列と、ユーザーから入力された自然言語指示を、それぞれテキスト埋め込みモデルに入力し、高次元の数値ベクトルに変換する。各 VC のベクトル化は、VC が追加・更新された際にのみ実行すればよく、計算結果を保存しておくことで、2 回目以降の実行が高速になる。

(3) **類似度計算と選出**:
以上で得られたユーザーの自然言語による指示のベクトルと、各 VC のベクトルとの間で、それぞれコサイン類似度を算出する。コサイン類似度は、ベクトル間の角度に基づき意味的な近さを測る指標であり、値が 1 に近いほど関連性が高いことを示す。算出された類似度スコアに基づき、上位 k 件の VC をフィルタリング結果として選出する。本研究では、実験的に $k = 3$ と設定した。

3.2 性能評価

フィルタリング性能を定量的に評価するため、評価用のデータセットを独自に構築し、複数のテキスト埋め込みモデルの評価実験を行った。

3.2.1 実験設定

本評価では、2 種類のテキスト埋め込みモデルを対象として性能比較を行った。ユーザーのローカル環境での動作を想定したオープンソースモデルとして、Sentence Transformer[7] ライブラリで利用可能な `all-mpnet-base-v2`[8] と、その軽量版である `all-MiniLM-L6-v2`[9] の 2 つを選定した。また、参考値として、OpenAI が提供する API サービスである `text-embedding-3-small`[10] も評価に加えた。

3.2.2 データセット

評価には独自に構築したデータセットを用いた。本データセットは、約 120 種類の架空の VC からなる VC 群と、約 140 件の自然言語の指示文とそれに対応する正解の VC のペアから構成される。正解の VC は、1 個から 3 個まで複数ある場合がある。正解の VC の具体例として、`I want to show my diplomas.` という指示文が与えられた場合は、VC 群の中の高校、大学、修士の卒業証明書 VC が正解データとして与えられる。

3.2.3 評価指標

本フィルタリングの目的は、ユーザーが先述した通り、意図した VC を網羅的に選出することである。したがって、その網羅性を評価するため、本研究では以下の 2 つの指標を導入した。

- **Recall@k**:
フィルタリングが「正解となる VC を、どれだけ漏らさずに拾えているか」を示す指標である。例えば、ある指示に対する正解 VC が 3 つ（A, B, C）あった際に、フィルタリング結果の上位 3 件（$k = 3$）に、そのうちの 2 つ（A, B）が含まれていた場合、この指示に対する Recall@3 は $2/3 = 66.7\%$ となる。本評価では $k = 3$ として算出している。
- **All-Collect@k**:
フィルタリング結果の上位 k 件に、クエリが要求する全ての正解 VC が含まれているクエリの割合である。例えば、あるクエリの正解 VC が 2 つである場合、上位 k 件にその両方が含まれていなければ、このクエリは達成できていないと見なす。本評価では $k = 3$ として算出している。

3.3 実験結果と考察

本実験では、先述した 2 種類のローカル環境で動作可能な埋め込みモデルの性能を比較評価した。また、参考値として、OpenAI 社の API モデルである `text-embedding-3-small` の結果も併記する。表 1 に、各モデルが出力するベクトルの次元数と評価結果を示す。

表 1　テキスト埋め込みモデルの性能比較 ($k = 3$)

テキスト埋め込みモデル	次元数	All-Collect	Recall	時間 (s)
`all-MiniLM-L6-v2`	384	0.8116	0.8841	0.016
`all-mpnet-base-v2`	768	0.8188	0.8720	0.054
`text-embedding-3-small` (参考)	1536	0.8841	0.9336	0.555 (通信時間を含む)

参考値として計測した `text-embedding-3-small` は、1536 という大きなベクトル次元数を持ち、All-Collect@3、Recall@3 の両指標で最も高い精度を示した。

本実験の主目的であるローカルモデルでは、処理が軽量で出力次元数の小さい `all-MiniLM-L6-v2` でも、実用上十分なフィルタリング精度が得られることがわかった。特に、384 次元のベクトルを生成する `all-MiniLM-L6-v2` は、より大規模なローカルモデルと同等の精度を出し、より高速に実行できた。OpenAI のより大規模なモデルには精度が劣るものの、この軽量性と高い性能は、モバイル端末のようなリソースに制約のある環境での動作も十分可能であり、本システムのフィルタリング機構として活用できると判断できる。

4. LLM によるクエリ生成

本節では、3 節で選出された複数の VC とユーザーの自然言語指示から、クエリ言語である DCQL のクエリを生成する手法について記述する。

本研究では、このクエリ生成を実現するため、大規模言語モデル（LLM）のファインチューニングによるアプローチを採用する。具体的には、オープンソースとして公開されている複数の LLM を基盤とし、DCQL 形式のクエリを生成するように追加学習を行った。

その際、学習を効率的に進めるため、PEFT (Parameter-Efficient Fine-Tuning) [11] の一種である LoRA (Low-Rank Adaptation) [12] を導入した。これは、元のモデルが持つ大部分の重みを固定したまま、ごく一部のパラメータのみを学習させる手法である。

さらに、4 ビット量子化技術 [13] を導入した。これは、モデルのパラメータを、通常よりも情報量の少ない 4 ビットの数値で近似的に表現する技術である。これにより、モデル全体のサイズが大幅に圧縮され、推論処理の高速化にも寄与する。

4.1 データセット

LLM のファインチューニングおよび評価には、独自に構築したデータセットを用いた。各データは、入力となるプロンプトと、正解となる DCQL クエリのペアで構成される。以下に、その一例を示す。

入力プロンプト

```
Given the following Verifiable Credentials
and a natural language query, generate
    a DCQL query to retrieve the
    requested information.
Available Verifiable Credentials:
VC 1: {
  "@context: [...],
  "id":"vc-hs-01",
  "type":["VerifiableCredential",
          "HighSchoolDiplomaCredential
             "],
  "issuer":
        {"name":"Tokyo Example High
           School"},
  ...
}
VC 2: {
  "@context: [...],
  "id":"vc-univ-02",
  "type":["VerifiableCredential",
          "UniversityDegreeCredential
             "],
  "issuer":{"name":"Example University
     "},
  "credentialSubject":{
    "studentName":"Taro Yamada",
    "major":"Computer Science",
    ...
  }
}
Natural Language Query:
I want to prove my major from my
    university degree.
```

期待される出力

```
{
  "credentials": [
    {
      "id": "university_degree_major",
      "format": "ldp_vc",
      "meta": {
        "type_values": [
```

```
          [
            "VerifiableCredential",
            "UniversityDegreeCredential
                "
          ]
        ]
      },
      "claims": [
        {
          "path": [
            "credentialSubject",
            "major"
          ]
        }
      ]
    }
  ]
}
```

この例が示すように、入力プロンプトは主に以下の 3 つの要素で構成されている。

- **固定の指示文:** LLM に DCQL クエリの生成を指示する文章（例中の黒いテキスト部分）。
- **VC 候補群:** ユーザーが保有し、事前フィルタリングされた VC のリスト（例中の青いテキスト以降の部分）。
- **ユーザーの自然言語指示:** ユーザーからの自然言語による指示文（例中の赤いテキスト以降の部分）。

一方、期待される出力となる DCQL は、ユーザーの指示に基づき、どの VC の、どの属性を提示すべきかを指定するクエリである。

本データセットの設計における重要な点として、この正解クエリは、指示された必要最低限の属性のみを要求するよう意図的に作成した点である。これは、LLM が最低限の情報開示を支援し、ユーザーのプライバシーを最大限に保護するための設計である。

データセットはこれらのペアを約 1000 個用意し、LLM にはこのペアを多数学習させる。このプロセスを通じて、LLM は与えられたユーザーの指示を正確に解釈し、適切な情報を特定して、定められた形式で出力するという、一連の推論能力を獲得する。

4.2 性能評価

複数の LLM を用いて、本手法の性能評価を行った。

4.2.1 実験設定

本評価では、14 種類のモデルを対象に、DCQL 生成タスクにおけるファインチューニングを実施した。

評価指標は以下の通りである。

- **精度**: 生成された DCQL が完全に正解と一致した割合。
- **平均推論時間**: 1 サンプルあたりの処理時間。

4.3 実験環境

本評価実験は、以下の環境で実施した。

- **CPU**: AMD Ryzen 7 9800X3D 8-Core Processor (16 threads)
- **Memory**: 92GiB
- **GPU**: NVIDIA GeForce RTX 4070

4.3.1 実験結果と考察

DCQL クエリ生成タスクにおける各モデルの性能評価結果を表 2 に示す。

表 2 DCQL 生成タスクにおける性能評価

モデル	パラメータ数	精度 (%)	推論時間 (s)
qwen3-4b[1]	4B	98.0	4.87
gemma-2-2b-it[2]	2B	96.0	10.24
gemma-2-2b[2]	2B	94.0	12.79
phi2[3]	2.7B	90.0	9.20
gemma-2b[2]	2B	86.0	3.24
deepseek-coder-1.3b[4]	1.3B	82.0	4.45
tinyllama[5]	1.1B	82.0	3.51
gemma-2b-it[2]	2B	80.0	3.02
qwen2-1.5b[1]	1.5B	78.0	11.93
gemma-3-1b-it[2]	1B	70.0	15.13
qwen2-0.5b[1]	0.5B	62.0	5.88

[1] Qwen (Alibaba Cloud) https://huggingface.co/Qwen
[2] Gemma (Google) https://deepmind.google/models/gemma/
[3] Phi-2(Microsoft) https://huggingface.co/microsoft/phi-2
[4] DeepSeek Coder (DeepSeek AI) https://huggingface.co/deepseek-ai/deepseek-coder-1.3b-base
[5] TinyLlama: https://huggingface.co/TinyLlama

実験結果から、qwen シリーズや gemma-2 シリーズなどをファインチューニングしたモデルは、非常に高い精度を示した。そして、モデルのパラメータサイズと DCQL 生成の精度には、概ね正の相関が確認された。

全体として多くのモデルで高い性能が確認できたことから、ローカル環境で動作する LLM であっても、実用上、問題なく DCQL クエリを生成できることが示唆される。

5. モバイル端末での実装

前章までで提案した、自然言語による指示から VC を提示するシステムを、モバイル端末上で動作するプロトタイプ実装を行った。VC は個人のスマートフォンやモバイルデバイス上のウォレットアプリケーションで管理されることが望まれており、モバイル環境での動作を検証することは、本研究の有効性を示す上で重要である。

5.1 実装概要

本プロトタイプは、以上で述べた一連のフローをモバイ

ルアプリケーションとして実装したものである。開発には以下の環境およびモデルを採用した。

実行環境　iPhone 16 Pro

開発言語　Swift

搭載モデル　第 4.2 章の評価に基づき、精度が良く、Swift からモデルを実行するライブラリがあることから **Gemma-2-2B-IT** を採用した。

利用ライブラリ　フィルタリングのためのテキスト埋め込みには、Apple の **NaturalLanguage**[14] を利用した。LLM による DCQL 生成には、Apple の機械学習用フレームワークである **CoreML**[15] を利用した。

5.2　実行例

図 3 は、実装したアプリケーションの動作画面である。ユーザーが `Show my health insurance but hide insurance number` と入力すると、システムが関連する VC を特定し、その内容を提示する様子を示している。実際にフィルタリングでは、`HealthInsuranceCardCredential`（該当の VC）と `HealthCheckCertificate`（近しい健康診断の VC）が選ばれ、LLM により、DCQL が生成された。

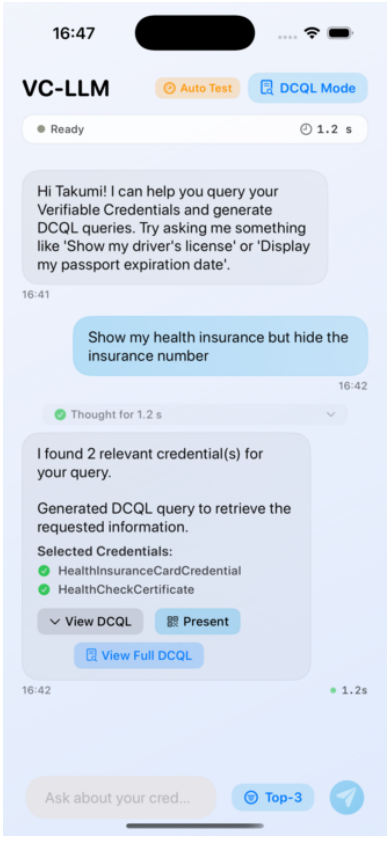

図 3　モバイルアプリケーションの実行画面

6.　まとめ

本稿では、多様な Verifiable Credentials を保有するユーザーが、自然言語で意図を指示するだけで、目的に合致した VC と属性を自動的に抽出し、提示用の Verifiable Presentation を生成するシステムを提案した。本システムの核心は、ユーザーのローカル環境で動作するファインチューニング済みのオープンソース大規模言語モデル (LLM) にあり、プライバシーを保護しつつ VC の利便性を向上させる点に特徴がある。

提案システムは、(1) Embedding モデルを用いた VC フィルタリング、(2) LLM による DCQL クエリ生成、(3) VP 生成、という 3 段階の処理フローで構成される。

さらに、本提案の有効性を示すため、提案システムをモバイルアプリケーションとして実装し、iPhone 上で一連の処理が完結する実装を行った。このプロトタイプでは、機微な個人情報を含む VC を外部に送信することなく、手元のデバイスで安全かつ効率的に利用できる。

参考文献

[1] M. Sporny, T. Thibodeau Jr, I. Herman, G. Cohen, M. B. Jones. Verifiable Credentials Data Model v2.0. `https://www.w3.org/TR/vc-data-model-2.0/`, May 2025.

[2] O. Terbu, T. Lodderstedt, K. Yasuda, D. Fett, J. Heenan. OpenID for Verifiable Presentations 1.0. `https://openid.net/specs/openid-4-verifiable-presentations-1_0.html`, July 2025.

[3] D. Fett, K. Yasuda, B. Campbell. Selective Disclosure for JWTs (SD-JWT). `https://datatracker.ietf.org/doc/draft-ietf-oauth-selective-disclosure-jwt/`, June 2025.

[4] ISO/IEC. ISO/IEC 18013-5:2021 Personal identification — ISO-compliant driving licence — Part 5: Mobile driving licence (mDL) application. `https://www.iso.org/standard/69084.html`, September 2021.

[5] D. Buchner, B. Zundel, M. Riedel, K. H. Duffy. Presentation Exchange v2.1.1. `https://identity.foundation/presentation-exchange/spec/v2.1.1/`, April 2024.

[6] S. Harris, A. Seaborne. SPARQL 1.1 Query Language. `https://www.w3.org/TR/sparql11-query/`, March 2013.

[7] UKPLab and Hugging Face. Sentence Transformers (SBERT) Documentation. `https://www.sbert.net/`, 2025. Accessed: 2025-08-12.

[8] sentence-transformers. all-mpnet-base-v2. `https://huggingface.co/sentence-transformers/all-mpnet-base-v2`, 2025. Accessed: 2025-08-12.

[9] sentence-transformers. all-MiniLM-L6-v2. `https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2`, 2025. Accessed: 2025-08-12.

[10] OpenAI. text-embedding-3-small. `https://platform.openai.com/docs/models/text-embedding-3-small`, 2025. Accessed: 2025-08-11.

[11] Sourab Mangrulkar and Sylvain Gugger and Lysandre Début and Younes Belkada and Sayak Paul and Benjamin Bossan and others. PEFT: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`, August 2025. Latest release: 0.17.0 on August 1, 2025.

[12] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large lan-

guage models. *arXiv preprint arXiv:2106.09685*, 2021.
[13] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms, 2023.
[14] Apple Inc. Natural language framework. `https://developer.apple.com/documentation/naturallanguage`, 2025. Accessed: 2025-08-19.
[15] Apple Inc. Core ml framework. `https://developer.apple.com/documentation/coreml`, 2025. Accessed: 2025-08-19.