Vol.2021-DPS-186 No.10
Vol.2021-CSEC-92 No.10
2021/3/15

ブラウザフィンガープリントを用いた会員ID 推定を提供する Web API システムの提案と構築

渡名喜　瑞稀[1,a)]　藤井 達也[1]　神 章洋[1]　利光 能直[2]　北條 大和[2]　齋藤 孝道[1]

概要：本論文では，ブラウザフィンガープリンティングを用いて，会員制 Web サイトの会員 ID 推定を行う Web API システムの構築を行った．会員 ID 推定を Web API 形式に対応したシステムとして構築したことにより，外部サービスと連携できるようにした．それによって，会員 ID 推定を用いることで精度向上ができると考えられる，リスクベース認証や不正検知への活用を図った．また，システムを Amazon Web Service が提供するサーバレスサービスを複数用いて構築した．サーバレスサービスを用いて構築することによって，リクエスト増大に伴うスケーリングやサーバ管理を容易にでき，随時リクエストを受け入れることができるようにした．本論文では，システムの構築に加え，システムの改善すべき点や改良できる点，システムを運用する際に直面する課題について考察した．

Proposal and construction of a Web API system that provides member ID estimation using browser fingerprints

Abstract: In this paper, we have developed a Web API system for estimating member IDs for membership websites using browser fingerprinting. We constructed a system that supports the Web API format for member ID estimation, so that it can be linked to external services. By doing so, we aimed to utilize the system for risk-based authentication and fraud detection, which are thought to improve accuracy by using member ID estimation. In addition, we constructed the system using several serverless services provided by Amazon Web Service. By using serverless services, the system can easily scale with the increase in requests and server management, and can accept requests at any time. In this paper, in addition to the construction of the system, we discuss the points that need to be improved, the points that can be improved, and the challenges that we face when operating the system.

1. はじめに

Web サーバがユーザのアクセスデータに含まれる情報を用いてブラウザを識別するブラウザフィンガープリンティングと呼ばれる技術がある．藤井ら [1] は，ブラウザフィンガープリンティングを用いた会員制 Web サイトの会員 ID 推定を行った．会員 ID 推定とは，ログイン前のブラウザからのアクセスにおけるブラウザフィンガープリントにより，会員の ID を推定する手法である．結果として，87% の精度で会員 ID 推定を行うことができた．

本論文では，ブラウザフィンガープリントを用いた会員 ID 推定をより大規模に，かつ，より短時間で，リアルシステムと連携することを目的とした，Web API システムの設計と構築を行った．

本研究の目的は，当該システムを用いて，会員 ID 推定を容易に行うことによって，リスクベース認証や不正端末検知など，リアルシステムにおける活用を目指すものである．

2. 関連知識

本章では，本論文で構築したシステムに用いた技術について説明をする．

2.1 Web Application Programming Interface

Web Application Programming Interface（以降 Web API）とは，アプリケーションを利用する際の Web 上のインターフェースである．ユーザがアプリケーション

[1] 明治大学
Meiji University
[2] 明治大学大学院
Graduate School of Meiji University
[a)] ee177100@meiji.ac.jp

サーバに設置した Web API に対して，HTTP リクエストを送信すると，サーバ内での処理結果が HTTP レスポンスとしてユーザに返される．

また，本論文では Web API とサーバ内のアプリケーションをまとめて Web API システムと呼び，構築した Web API システムへ送信する HTTP リクエストを API リクエスト，API システムから送信される HTTP レスポンスを API レスポンスと呼ぶ．

2.2 Amazon Web Service

本節では，Amazon Web Service（以降 AWS）について，また本論文で用いた AWS サービスについて記述する．

AWS は，世界で最も包括的で広く採用されているクラウドプラットフォーム．AWS には，コンピューティング，ストレージ，データベースなどのインフラストラクチャテクノロジーから機械学習，人工知能，データレイクと分析，IoT などの新しいテクノロジーに至るまで，他のどのクラウドプロバイダーよりもはるかに多くのサービスがあり，サービス内にはさらに多くの機能がある [2]．

2.2.1 Amazon API Gateway

Amazon API Gateway（以降 API Gateway）は，API の作成，公開，保守，モニタリング，保護を行えるサービス．ユーザは API Gateway を用いることで最大数十万規模の同時 API コールの受け入れと処理に伴うすべてのタスクを取り扱うことができる [3]．

2.2.2 AWS Lambda

AWS Lambda（以降 Lambda）は，サーバーレスコンピューティングサービス．ユーザは Lambda を用いることで，サーバのプロビジョニングや管理などを行わずにコードを実行できる．Lambda は自動的，かつ正確にコンピューティング実行能力を割り当て，受信リクエストやイベントに基づいたコードを実行する [4]．

2.2.3 Amazon Elastic Container Ragistry

Amazon Elastic Container Registry（以降 ECR）は，完全マネージド型のコンテナレジストリ．このレジストリを使うと，コンテナイメージとアーティファクトをどこにでも簡単に保存，管理，共有，デプロイできる [5]．

2.2.4 Amazon DynamoDB

Amazon DynamoDB（以降 DynamoDB）は，規模に関係なく数ミリ秒台のパフォーマンスを実現する，key-value およびドキュメントデータベース．DynamoDB は，サーバレスアプリケーションであり，ユーザのプロビジョニングや管理，保守，運用などをする必要がない [6]．

2.3 ブラウザフィンガープリンティング

ユーザが Web サイトにアクセスした際，ユーザが利用したブラウザの情報を取得することができる．取得した情報を利用して，ユーザの識別を行う技術をブラウザフィンガープリンティング（以降フィンガープリンティング）という．また，取得した情報をブラウザフィンガープリント（以降フィンガープリント）という．

Eckersley ら [7] は，フィンガープリントを採取するサイトを構築し，94.2%のフィンガープリントがユニークであることを示した．

3. 提案 Web API システム

本節では，提案 Web API システムの詳細について説明する．

3.1 提案 Web API システム概要

提案 Web API システムは AWS が提供するサービスを用いて構築をした．

使用した AWS サービスと構成要素内でのリソース名を**表 1** に示す．

表 1 使用した AWS サービスとリソース名

サービス名	リソース名
API Gateway	API Gateway_1
Lambda	Lambda_1,Lambda_2,Lambda_3
DynamoDB	DynamoDB_1
ECR	ECR_1

3.2 AWS サーバレスサービス利用目的

提案 Web API システムを AWS が提供するサーバレスサービスを用いて実装した．その目的を以下に示す．

- リクエストが集中した際，自動スケーリングをシンプルに実現できる（サーバの CPU やメモリやストレージの量や数を増強することが不要）
- リクエストを常に受け入れるためのサーバ管理を排除する

3.3 提案 Web API の利用

API のユーザは，提案 Web API を利用した会員 ID 推定を以下の手順で行う．

(1) 会員 ID 推定依頼 API リクエストを送信する
(2) 会員 ID 推定結果取得 API リクエストを送信する

会員 ID 推定依頼 API リクエストに含む情報を**表 2**，会員 ID 推定依頼 API レスポンスに含む情報を**表 3** に示す．

また，表 2 をまとめてアクセスデータと呼ぶ．

表 2 会員 ID 推定依頼 API リクエストに含む情報

項目	説明
IP アドレス	推定対象端末の IP アドレス
アクセス時刻	推定対象端末がサーバにアクセスした時刻
User-Agent 文字列	推定対象端末の User-Agent 文字列

会員 ID 推定結果取得 API リクエストに含む情報を**表 4**，

Vol.2021-DPS-186 No.10
Vol.2021-CSEC-92 No.10
2021/3/15

表 3 会員 ID 推定依頼 API レスポンスに含む情報

項目	説明
推定結果取得用キー	推定結果を取得する一意の文字列

会員 ID 推定結果取得 API レスポンスに含む情報を**表 5** に示す.

表 4 会員 ID 推定結果取得 API リクエストに含む情報

項目	説明
推定結果取得用キー	推定結果を取得する一意の文字列

表 5 会員 ID 推定結果取得 API レスポンスに含む情報

項目	説明
推定状況	会員 ID 推定の状況
推定結果	会員 ID 推定の結果

会員 ID 推定結果取得 API レスポンスに含む推定状況には成功，処理中，エラーの 3 種類存在する．**表 6** に 3 種類の推定状況を説明する.

表 6 推定状況

推定状況	説明
成功	会員 ID 推定処理が終了
処理中	会員 ID 推定処理中
エラー	**表 7** に記述する特徴点のいずれかが作成できないので，推定不可能

表 7 生成できない場合，会員 ID 状況がエラーとなる特徴点

生成する特徴点
ISP 名
都市名
OS バージョン
ブラウザバージョン
OS メジャーバージョン

それぞれの推定状況に対する推定結果項目を以下に説明する.

(1) 推定状況が成功：5.5 で記述する会員 ID なし推定の結果に対応する
  (a) 会員 ID が存在すると推定：推定された ID と推定の尤度
  (b) 会員 ID が存在しないと推定：空文字列
(2) 推定状況が処理中：空文字列
(3) 推定状況がエラー：空文字列

よって API のユーザは，会員 ID 推定結果取得 API レスポンスを閲覧することにより，推定対象のアクセスデータの会員 ID 推定結果がわかる.

Web API の利用についての概要図を**図 1** に示す.

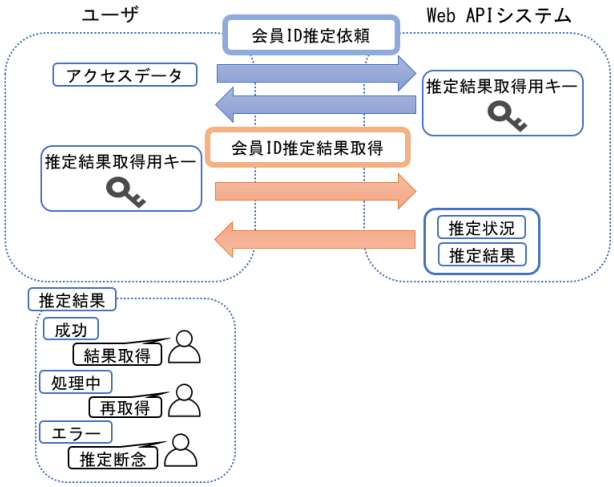

図 1 Web API 利用概要図

3.4 提案 Web API 構成要素

本節では提案 Web API の構成要素について説明する．全体図を**図 2** に示す.

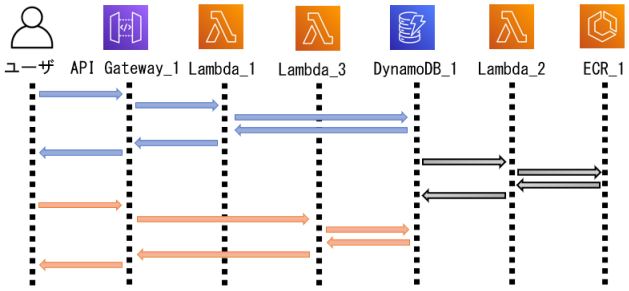

図 2 Web API 全体図

以下に図 2 に示した構成要素それぞれの処理を示す.

3.4.1 API Gateway_1

ユーザから API リクエストを受け取り，リクエストに応じた API レスポンスを返す.

API Gateway_1 には，会員 ID 推定依頼と会員 ID 推定結果取得の 2 種類の API リクエストが送信される.

以下にそれぞれの API リクエストが送信された際の処理を示す.

- 会員 ID 推定依頼：IP アドレス，アクセス時刻，User-Agent 文字列を受け取り，Lambda_1 を起動する．Lambda_1 での処理後，推定結果取得用キーを返す
- 会員 ID 推定依頼：推定結果取得用キーを受け取り，Lambda_3 を起動する．Lambda_3 での処理後，推定状況と推定結果を返す

3.4.2 Lambda_1

API Gateway_1 が会員 ID 推定依頼 API リクエストを受け取った際に，起動される.

以下に処理を示す.

(1) 一意の値をとる推定結果取得用キーを作成する
(2) DynamoDB_1 に，API リクエストに含まれるアクセスデータ，(1) で作成した推定結果取得用キー，表 3 で示した処理状況を処理中，推定結果を空文字列で記

録する．

また，推定結果取得用キーを DynamoDB_1 のプライマリキーとする

(3) 推定結果取得用キーを API Gateway_1 を介して，ユーザにレスポンスとして返す

3.4.3 DynamoDB_1

1 アクセスデータに対して，推定結果取得用キー，アクセスデータ，表 3 で示した処理状況，推定結果が記録される．

3.4.4 Lambda_2

DynamoDB_1 の変化が起きた際に起動される．

Lambda_2 は，DynamoDB_1 内の変化すべてに対し起動してしまうため，DynamoDB_1 内にデータが追加された場合のみ処理を行うようにする．

また，Lambda_2 で取り扱うコードやデータは ECR 上に用意し，ECR から Lambda_2 にデプロイがされる．

Lambda_2 では，以下に示す会員 ID 推定処理を行う．

(1) DynamoDB_1 内に新たに記録された，アクセスデータと推定結果取得用キーを取得する
(2) アクセスデータに対して 4.2 で記述する特徴点追加を行う
(3) アクセスデータに対し，4.3 で記述する統計量を用いた特徴点を作成し，追加する
(4) 取得したアクセスデータに対し，5.4 で記述する会員 ID 推定を行う
(5) 取得した推定結果取得用キーを用い，DynamoDB_1 内の該当データの会員 ID 推定処理状況を処理完了，会員 ID 推定結果を推定結果に更新する

3.4.5 Lambda_3

API Gateway_1 が会員 ID 推定結果取得リクエストを受け取った際に，起動する．

リクエストに含まれる推定結果取得用キーを用いて DynamoDB_1 から対象の項目を取得する．その後，項目から推定状況と推定結果を取得し，API Gateway_1 を介し，ユーザにレスポンスとして返す．

4. 会員 ID 推定データ

本節では，会員 ID 推定に用いるデータについて示す．

4.1 アクセスデータが持つ特徴点の詳細

会員 ID 推定に用いるアクセスデータはそれぞれ，**表 8** に示す特徴点と会員 ID を持つ．

表 8 アクセスデータが持つ特徴点の例

特徴点	例
タイムスタンプ	2019-07-10 07:11:14
User-Agent 文字列	Mozilla/5.0 (Android 5.1.1; Tablet; rv:68.0) Gecko/68.0 Firefox/68.0
IP アドレス	192.168.100.1

4.2 アクセスデータから新たな特徴点を生成

表 8 で示した特徴点の値を用いて新たな特徴点を生成した．**表 9** に，生成した特徴点を示す．

表 9 アクセスデータから生成した特徴点

元の特徴点	生成した特徴点
タイムスタンプ	年, 月, 日, 時, 分, 秒, 曜日，UNIX 時間形式のタイムスタンプ
IP アドレス	第 1 オクテット，第 2 オクテット，第 3 オクテット，第 4 オクテット
IP アドレス	ISP 名
IP アドレス	国名，都市名，市区町村名，緯度，経度
User-Agent 文字列	OS 名, OS のバージョン, Web ブラウザ名, Web ブラウザのバージョン, 機種名, 機種のブランド名，OS のメジャー・マイナー及びメンテナンスバージョン Web ブラウザのメジャー・マイナー及びメンテナンスバージョン

OS 名，OS バージョン，Web ブラウザ，Web ブラウザのバージョンは，user-agents2.0[8] を用いて，User-Agent 文字列から抽出した．ISP 名は，pyisp[9] を用いて，IP アドレスから取得した．国名，都市名，市区町村，緯度，経度は GeoIP2[10] を用いて，IP アドレスから取得した．GeoIP2 は MaxMind 社の GeoLite2 のデータベースを使用するライブラリであり，GeoLite2 に保存されている位置情報は過去のある時点での情報である．

4.3 統計量を計算した特徴点を生成

4.2 で作成したデータの一部特徴点を用いて，統計量を以下の手順で計算し，新たな特徴点とする．

(1) 文字列の特徴点を連結することで，新たな特徴点（以降，連結特徴点と呼ぶ）を生成（**表 10**）
(2) 連結特徴点の値が同じであるアクセスデータの集合（以降，連結特徴点同一集合と呼ぶ）を作成（**表 11**）
(3) 連結特徴点同一集合において，**表 12** に示す特徴点 X と特徴点 Y の**表 13** に示す統計量を計算

表 10 生成した連結特徴点

名前	連結する特徴点
連結特徴点 A	User-Agent 文字列，都市名，ISP 名
連結特徴点 B	User-Agent 文字列，ISP 名
連結特徴点 C	User-Agent 文字列，都市名

表 11 作成した連結特徴点同一集合

名前	対象の連結特徴点
連結特徴点 A 同一集合	連結特徴点 A
連結特徴点 B 同一集合	連結特徴点 B
連結特徴点 C 同一集合	連結特徴点 C

表 12 連結特徴点同一集合において統計量を計算する特徴点

連結特徴点同一集合	特徴点 X	特徴点 Y
連結特徴点 A 同一集合	時刻 (hour)，曜日	なし
連結特徴点 B 同一集合	時刻 (hour), 曜日, 緯度, 経度	都市名
連結特徴点 C 同一集合	時刻 (hour)，曜日	ISP 名

表 13 計算する統計量およびその対象の特徴点

計算する統計量	計算対象
連結特徴点同一集合の要素数	特徴点 X，特徴点 Y
平均値，標準偏差，四分位数	特徴点 X
平均値および中央値との差	特徴点 X
平均値からどの程度離れているのかを表す特徴	特徴点 X
最頻値	特徴点 Y
最頻値の出現回数	特徴点 Y
最頻値と値が一致するかどうか	特徴点 Y
計算対象の特徴点の値の種類	特徴点 Y

表 13 で示した，”平均値からどの程度離れているのかを表す特徴”は，計算対象の特徴点の値が [平均値 − 標準偏差, 平均値 + 標準偏差] および [平均値 − 2×標準偏差, 平均値 +2×標準偏差] の範囲に入っているかどうかをそれぞれ True または False で表す特徴点である．

5. 会員 ID 推定

本節ではアクセスデータをベクトルデータに変換する方法，会員 ID 推定の方法を示す．

5.1 ベクトルデータ作成

アクセスデータをベクトルデータに変換する手順を示す．

提案システムでは，任意の 2 つのアクセスデータを組み合わせ，ベクトルデータを作成した．

提案システムに使用するベクトルデータの作成手順を以下に示す．

(1) 組み合わせる 2 つのアクセスデータに，4.2，4.3 の方法で特徴点を追加する
(2) 2 件のアクセスデータを組み合わせ，特徴点の値を比較した情報を新たな特徴点として追加する
(3) 組み合わせたデータに対し，**表 14** に記述する数値化を行う
(4) 5.1.1 に示す方法で，正解ラベルを付加

特徴点を比較した情報を表す特徴点を**表 15** に示す．

表 14 ベクトルデータの数値化

数値化する方法	対象の特徴点
ハッシュ化することで数値化する	値が文字列の特徴点
値を 0 または 1 で表す	値が真偽値の特徴点
値をそのまま使用する	値が数値の特徴点

表 15 比較した情報を表す特徴点

比較した特徴点	比較した情報を表す特徴点
値が文字列の特徴点，値が真偽値の特徴点	値の一致有無
値が数値の特徴点	値の差
緯度および経度	2 件のアクセスデータの位置情報の直線距離
直線距離およびタイムスタンプ	2 件のアクセスデータの距離を移動する場合のタイムスタンプの差から計算する速度
OS バージョンおよびタイムスタンプ	OS のバージョンとアクセス時刻の前後関係に矛盾があるかどうか

5.1.1 正解ラベルの作成

正解ラベルを作成する方法を示す．

データセットのアクセスデータは，会員ごとに異なる固有の会員 ID を持つ．それを用いて，正解ラベルは，5.1 の方法で作成する 2 つのアクセスデータの会員 ID が同一性を表す値とした．ラベルの値は同一であれば 1，同一でなければ 0 とした．

5.2 ニューラルネットワークの構成

本論文で作成したニューラルネットワークの構造を**表 16** に示す．また，各層で BatchNormalization を行い，損失関数は BinaryCrossentropy，最適化関数には Adagrad（パラメータはデフォルトの値）を用いた．すべての実験でバッチサイズは 2,000 を用いた．

表 16 ニューラルネットワークの構造

層	ユニット数	活性化関数	DropOut
入力層	特徴点の数	relu	無し
中間層 1	500	relu	0.3
中間層 2	500	relu	0.3
中間層 3	500	relu	0.3
中間層 4	50	relu	0.3
出力層	2	sigmoid function	0.3

5.3 モデルの作成

推論モデルの作成について示す．Web API システムに用いる推論モデルは，会員 ID 推定を行う前に作成する必要がある．

作成は以下の手順で行った．

(1) 会員 ID 付フィンガープリントからアクセスデータをランダムに 2 つ抽出する
(2) (1) で抽出した 2 つのアクセスデータを組み合わせ，ベクトルデータを作成する
(3) (1)，(2) を約 2000 万回行い，作成をしたベクトルデータを用いて，推論モデルを作成する

作成したモデルは，ベクトルデータを構成する 2 つのアクセスデータの会員 ID が同一である尤度を出力する．

5.4 会員 ID 推定

会員 ID 推定について示す．データセットのアクセスデータは，会員ごとに異なる固有の会員 ID を持つ．

会員 ID 推定は，5.3 で作成した深層学習モデルを用いて，推定対象のアクセスデータが，どの会員 ID 付フィンガープリントの会員 ID と一致するかを推定することで実現した．

会員 ID 推定の方法を以下に示す．

(1) 推定対象アクセスデータを会員 ID 付フィンガープリント全てと組み合わせ，5.1 で記述する方法で，複数のベクトルデータを作成する
(2) それぞれのベクトルデータに対し，構成するアクセスデータの会員 ID が同一であるかを，5.3 の方法で作成したモデルを用いて推論を行い，尤度を取得する
(3) 5.5 に記述する会員 ID なし推定を行い，推定対象アクセスデータの会員 ID が会員 ID 付フィンガープリントに存在しないと推定された場合，会員 ID 推定を行わず，存在すると推定された場合には (4) を行う
(4) (2) で取得した尤度が大きい上位 3 つのベクトルデータから会員 ID 付フィンガープリントの会員 ID を取得し，出現頻度の最も高い会員 ID を推定結果とする．また，3 つの会員 ID の出現頻度が等しい場合，尤度が最も大きい会員 ID を推定結果とする

5.5 会員なし推定

会員 ID なし推定とは，推定対象のアクセスデータの会員 ID が，会員 ID 推定する際に用いる会員 ID 付フィンガープリントに存在するかを推定することを指す．以下に会員 ID なし推定の方法を示す．

(1) 会員 ID 推定を行う
(2) すべての会員 ID 推定結果の尤度が閾値以下であることを確認する
(3) 推定対象のアクセスデータは会員 ID 付フィンガープリントに一致する会員 ID が存在しないとする

推定対象のアクセスデータの会員 ID が，過去のアクセスデータに存在しない場合，正しい会員 ID を推定することができない．そのため，会員 ID なし推定を行うことで，正しい会員 ID が存在すると推定された場合のみ，会員 ID 推定を行うことによって，全体の精度向上を図った．

6. 提案 Web API システムの活用展望

本研究で構築した会員 ID 推定 Web API システムは API リクエストを送信することにより，会員 ID 推定を行うことができる．そのため，以下のような活用が見込まれる．

6.1 リスクベース認証

本研究で用いる会員 ID 推定はフィンガープリンティングの技術を用いている．そのため，Web サイトにログインしたユーザの利用端末情報やアクセスした IP アドレスなどが異なる際に追加の認証を行うリスクベース認証に用いることが見込まれる．

以下のような活用が考えられる．

(1) Web サイト管理者は，ユーザの Web サイトにログイン時に，サーバでアクセスデータを取得する
(2) Web サイト管理者は会員 ID 推定 Web API システムに，取得したアクセスデータを含むリクエストを送信する
(3) レスポンスを確認し，会員 ID 推定結果がユーザのログインしようとしている会員 ID と異なる場合，ユーザに対し追加の認証を行う

6.2 不正端末，Bot 検知

本研究で用いる会員 ID 推定の処理に 5.5 で記述した会員 ID なし推定がある．会員 ID 推定に用いる会員 ID 付フィンガープリントを，過去に不正を行ったと判別された会員や Bot と判別されたアクセスデータにすることで，不正端末の検知への利用が見込まれる．

以下のような活用が考えられる．

(1) ユーザが Web サイトにログインした際にサーバでアクセスデータを取得する
(2) Web サイト管理者は会員 ID 推定 Web API システムに，取得したアクセスデータを含むリクエストを送信する
(3) レスポンスを確認し，過去 Bot 判定されたアクセスデータセット内に同一端末があると推定された場合，Web サイトにアクセスしたユーザを Bot と判断する

7. 提案 Web API システムの課題

本論文で構築をした Web API システムでは解決をできなかった課題やシステム運用時の問題点への改善案，改良できる点を示す．

7.1 入力値の検証

本論文で構築した Web API システムに送信される API リクエストは，正しい形式の入力を前提にしている．そのため，我々が意図していない入力がされた場合，正しい処理が行われない．

例として，以下のような API リクエストは意図していない．

- 会員 ID 推定依頼 API に必要な情報が含まれない
- 会員 ID 推定結果取得 API に送られた推定結果取得用キーが不当な値である

したがって，会員 ID 推定処理が行われる前に入力値が正しい形式であることの確認，入力値の不備を API レスポンスによって知らせる設計が必要とされる．

7.2 API リクエストの最大値設定

本論文で構築をした Web API システムに送信される API リクエストは，そのすべてが会員 ID 推定処理されるようになっている．そのため，システム運用側の意図よりも多くのリクエストが送信される可能性がある．その結果，システムで処理できるリクエストの総数を超えてしまい，API のユーザがリクエストを送信してからレスポンスが返るまでの時間が大きくなる．

それを防ぐために，API のユーザが送ることができるリクエストの最大値を規定し，その値よりも多いリクエストが送られた際に，会員 ID 推定処理する前に遮断する処理が求められる．

7.3 会員 ID 推定総数の可視化

会員 ID 推定システムで用いる会員 ID 付フィンガープリントや推論モデルは，会員 ID 推定システムを利用するユーザ固有のものである．特定のユーザが他のユーザよりも ID 推定を多量に利用していると，システム全体の処理時間が大きくなる可能性がある．

そのため，ユーザごとに会員 ID 推定システムの利用総数をユーザ側，システム運用側の両方が確認できるようにする必要がある．

7.4 会員 ID 付フィンガープリントの巨大化への対応

会員 ID 推定システムを AWS が提供するサービスを用いて構築したため，会員 ID 推定数が増加した際に伴う，スケーリング問題の解決を図った．

しかし，会員 ID 推定を行う際に，会員 ID 付フィンガープリントを用いるため，会員 ID 付フィンガープリントが大きくなった場合，以下のような問題が起こり得る．

- 会員 ID 推定にかかる時間の増大化
- 会員 ID 付フィンガープリントが Lambda 上のメモリに乗らない

したがって，会員 ID 推定に用いる会員 ID 付フィンガープリントの大きさを処理に影響がない程度に小さく，会員 ID 推定精度に影響がない程度に大きくするような方法が必要である．

7.5 OpenAPI Specification による仕様の記述

本論文で構築した Web API システムに用いられるインターフェースは Web API 形式である．そのため，OpenAPI Specification[11] に従って API の仕様を記述することができる．

OpenAPI Specification に従って，API を記述することにより以下のような利点を得ることができる．

- API 定義のバージョン管理を効率化
- API 仕様の明確化
- Swagger UI[12] を用いての動作確認

8. まとめ

本論文では，アクセスデータを用いて会員 ID 推定を行う Web API システムの構築を行った．

Web API システムを AWS が提供するサービスを用いて構築した．それによって，自動的なスケーリングを可能にし，サーバ管理を容易にすることにより，大規模な会員 ID 推定を常に行うことを可能にした．

また，Web API 形式にしてインターフェースを定めたことにより，他サービスとの連携を行うことができるようにした．そのため，リスクベース認証や不正端末検知などへの活用が見込まれる．

参考文献

[1] 藤井 達也, 渡名喜 瑞稀, 利光 能直, 柴田 怜, 北條 大和, 齋藤 孝道, ”PC とモバイル端末における深層学習を用いた ID の推定手法の提案と実装”, コンピュータセキュリティシンポジウム 2020, 2020.
[2] Amazon Web Service, https://aws.amazon.com/jp/what-is-aws/
[3] Amazon API Gateway, https://aws.amazon.com/jp/api-gateway/
[4] AWS Lambda, https://aws.amazon.com/jp/lambda/
[5] Amazon ECR, https://aws.amazon.com/jp/ecr/
[6] Amazon DynamoDB, https://aws.amazon.com/jp/dynamodb/
[7] P. Eckersley, ”How Unique Is Your Web Browser?”, in Proc. of the 10th international conference on Privacy enhancing technologies (PETS’10), 2010.
[8] python-user-agents, https://github.com/selwin/python-user-agents
[9] pyisp, https://github.com/ActivisionGameScience/pyisp/
[10] GepIP2-python, https://github.com/maxmind/GeoIP2-python/
[11] OpenAPI,https://swagger.io/specification/
[12] Swagger UI, https://swagger.io/tools/swagger-ui/