多言語自動翻訳掲示板の利活用に関する実践研究

早稲田大学人間科学部人間情報科学科
西村研究室

学籍番号：1J20F037
名前：奥村飛悠

2023 年 12 月 5 日

目次

第 1 章

はじめに

1.1　現状

インターネットの普及やソーシャルメディアの台頭により，オンラインでのコミュニケーションが一般的になっている（大向，2006）. その中でも，誰でも気軽に参加することのできる掲示板は重要なコミュニケーションの場となっている. 日本では，”5 ちゃんねる”（サイト名を”2 ちゃんねる”から”5 ちゃんねる”へ 2017 年 10 月に変更）が広く知られている一方で，米国発の”reddit”は国際的に認知度が高く，日別のアクティブユーザー数は 5700 万人，総投稿数は 130 億投稿を超えている (Reddit Inc，2023). これらの大規模な掲示板は情報の集積場所として，またユーザー間の活発な議論の場として重要な役割を果たしている.

しかしながら，掲示板は誰もが利用できるコミュニケーションの場であるにも関わらず，現状ではそのコミュニケーションは主に同一言語間で行われている. 具体的には，”5 ちゃんねる”では主に日本語，”reddit”では主に英語が使用されている. そのため，異なる言語を使用するユーザーは，翻訳ツールや外部の翻訳サービスを頼るか，専用のスレッドや言語コミュニティを探すことが一般的となっている. しかし，これらの方法にはデメリットが存在する. 例えば，翻訳ツールを利用すると時間と手間がかかるため，掲示板の持つ即時性や気軽さというメリットを十分に享受することが難しくなる.

同一言語間でのコミュニケーションが主流となっている背後には複数の要因が考えられるが，その一つとして翻訳技術の品質が十分でなかったことが考えられる.2004 年には「コミュニケーションツールとして使用する場合に十分な品質を持っているとはいい難い」（船越ほか，2004）との指摘があり，さらに 2010 年にも「近年，翻訳技術は急速に進展しているが，高精度な翻訳を行うことは困難である．コミュニケーションにおいて，不適切な翻訳箇所を含む文章は話者間の相互理解を困難にし，円滑なコミュ

ニケーションの妨げとなる」（宮部ほか，2010）とも指摘されている. また，多言語間でのコミュニケーションにおいては，翻訳の品質が極めて大きな影響力を持つことも確認されている（船越ほか，2004）. このように，コミュニケーションに大きな影響を与える翻訳技術の品質が十分でなかったため，ユーザーは翻訳技術を利用して会話の中心となっている言語以外を使用してまで会話を試みなかったのではないだろうか.

1.2 翻訳技術の進歩

しかしながら，翻訳サービスの精度は日々向上している. これについて，「近年，Google 翻訳や DeepL，そしてページ全体翻訳機能の進化が著しい」(村本，2022) との報告があり，その背景には機械学習の進歩が影響を与えている.「Google 英日翻訳が NMT（ニューラル機械翻訳）を採用したことで，目標言語の流暢さが格段に向上した」(影浦，2017) との報告がある. 同様に NMT を採用している DeepL は，2017 年にサービスを開始し，その高品質な翻訳サービスが評価されている（亀田，2022）. さらに「2020 年と 2021 年には，文章の意味をより正確に伝えられ，業界特有の専門用語もうまく処理できる新たなモデルを発表」（DeepL，2023）している. これらのことから，翻訳サービスの精度は日々向上されていることが分かる.

また，多くの翻訳サービスが開発者向けに API を提供している. その代表例としては，Google Cloud の Translation AI（Google Cloud，2023）や DeepL API（DeepL，2023）がある. これらのサービスを開発者が利用するための便利なライブラリも存在している. 具体的には，Google Translate API（現在の Translation AI）を利用するための Python のライブラリである googletrans（PyPI，2023a）や deepl（PyPI，2023b）がある. このような API やライブラリの存在により，開発者は翻訳機能を自身のサービスに容易に組み込むことが可能となっている.

1.3 既存サービスと先行研究

過去には，”enjoy Korea”という日本語と韓国語の翻訳機能を持つ掲示板サービスが存在していたが，利用率の低下を理由に 2009 年 6 月 8 日にサービスを終了している（野津，2009）. また，小川ら（2009）は日本語とウイグル語間の翻訳掲示板システムを開発している. しかし，彼らの研究は主にシステムの開発に焦点を当てており，システムを使用するユーザーのデータ収集やその分析までには至っていない.

一方，藤井ら（2005）はアノテーションや折り返し翻訳に着目し，中国語，韓国語，日本語間の翻訳 BBS である”AnnoChat”を開発した. 翻訳の精度がコミュニケーションの理解度に影響を与える可能性を示しているが，ユーザー同士の具体的なコミュニケー

ションの内容までは調査していない. また，吉野ら（2006）はユーザインタフェースのカスタマイズ性に焦点を当てた研究を行い，”CustomChat”というシステムを開発したが，これも具体的なチャットの内容などについては触れられていない.

これらの事例や研究を見ると，多言語間のコミュニケーションを可能にする翻訳掲示板に関する研究やサービスは確かに存在している. しかし，それらは主にシステムの開発や翻訳の精度と理解度の関係性，ユーザインタフェースの改良に焦点を当てており，異なる言語を使用するユーザーがシステムをどのように使うのか，どのようなコミュニケーションが起こるのかという点については，まだ十分に研究されていないと言える.

1.4 本研究の概要と目的

これらの背景から，本研究では，掲示板のグローバル化を進めるため，近年の高精度な翻訳サービスを利用した多言語自動翻訳掲示板の開発とその利活用について実践的な研究を行う. 我々が提案する多言語自動翻訳掲示板では，ユーザーは表示言語を選択することにより，選択した言語で掲示板の投稿を閲覧することを可能にする機能をつける. これにより，異なる言語を使用するユーザー間でも，自由なコミュニケーションが促進され，掲示板の持つ即時性や気軽さというメリットを維持することができる.

そして，この掲示板をインターネット上に公開し，使用者から得られるデータを収集する. その後，得られたデータを分析し，多言語自動翻訳掲示板がユーザーのコミュニケーションにどのような影響を与えるのか，多言語自動翻訳掲示板上で異なる言語を使用するユーザー同士がどのようなコミュニケーションをするのかを評価する. 具体的には，ユーザー間のコミュニケーション量や内容，トピックの多様性，言語間のコミュニケーション方法などを指標として用いる.

我々の研究は，新たな掲示板の形を示すだけでなく，機械翻訳技術とその実用化の進歩に貢献することを期待している. 本研究の結果が，ユーザーが自由に多言語コミュニケーションを享受できるインターネットの環境整備に向けた一歩となることを願っている.

第 2 章

多言語自動翻訳掲示板について

2.1　システムの概要

本研究では，多言語自動翻訳掲示板である「The Channel」という Web アプリケーションを開発した. このシステムは，世界中のユーザーが自分の言語で投稿することができる. そして，その投稿はユーザーが選択した言語に翻訳されて表示されることで，ユーザーは好きな言語でコンテンツを読むことができる.

システムの中核は，機械翻訳技術が担っている. これにより，ユーザーが投稿したテキストはリアルタイムで他の言語に翻訳され，多様なユーザーがアクセスできるようになる. 例えば，日本語で書かれた投稿は，英語，スペイン語，中国語などに瞬時に翻訳され，異なる言語のユーザー間の交流を可能にする.

システムは，掲示板として必要最低限の機能を備えている. ユーザーは簡単にスレッドを作成することや，自分の母国語でコメントを投稿することができる.

セキュリティに考慮し，ユーザーの個人名やメールアドレス等を取り扱わないようにしている.

2.2　主な機能と利用フロー

掲示板は以下の機能を持つ.

1. スレッド作成
   スレッドの作成を行う.
2. コメント投稿
   スレッドに対して，コメントを投稿する.
3. 閲覧

スレッドに投稿されたコメントの閲覧.

4. 言語選択
   閲覧する言語を選択する.

以下にシステム利用フローを示す.

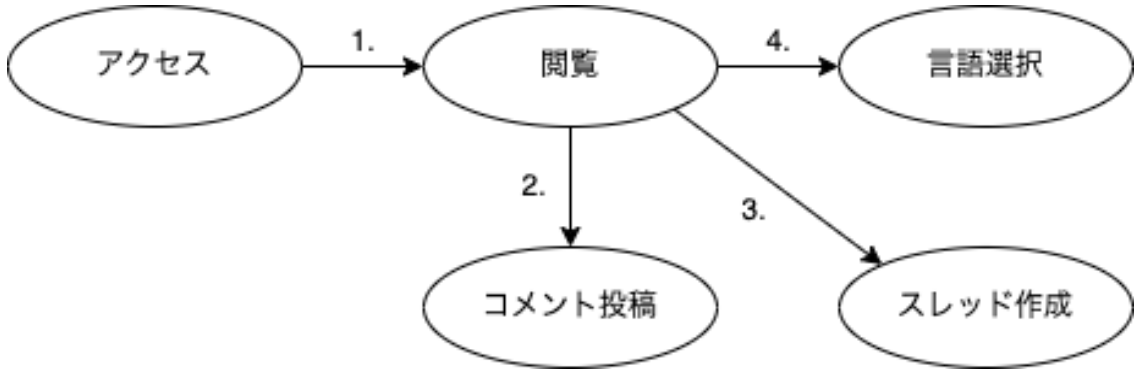
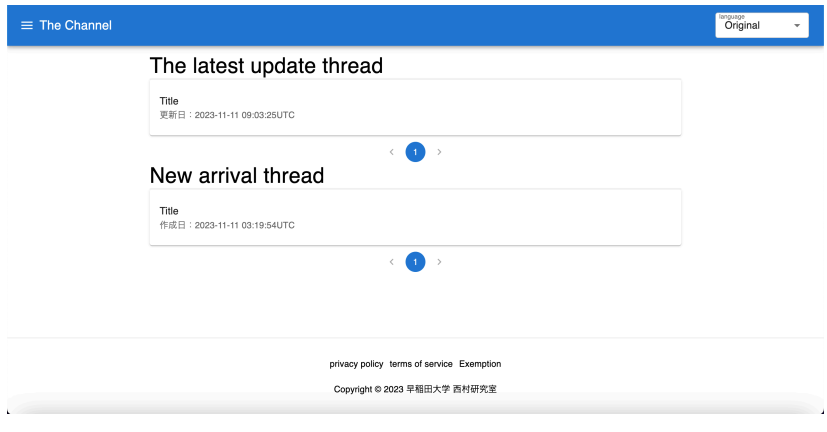


図 2.1: システム利用フロー

1. ユーザーはブラウザを通して掲示板を閲覧する.
2. ユーザーは閲覧しているスレッドに対してコメントを投稿することができる.
3. ユーザーは自由にスレッドを作成することができる.
4. ユーザーはいつでも言語を選択することができる.

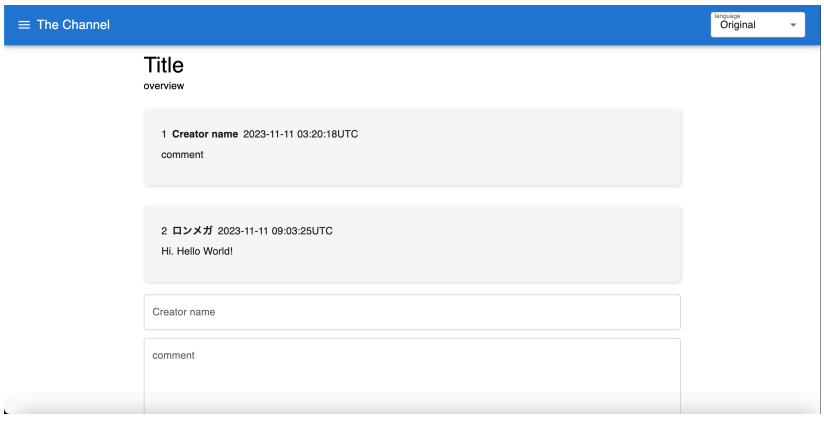
2.3 機能詳細と主な画面操作

主な画面のフローを以下に示す.

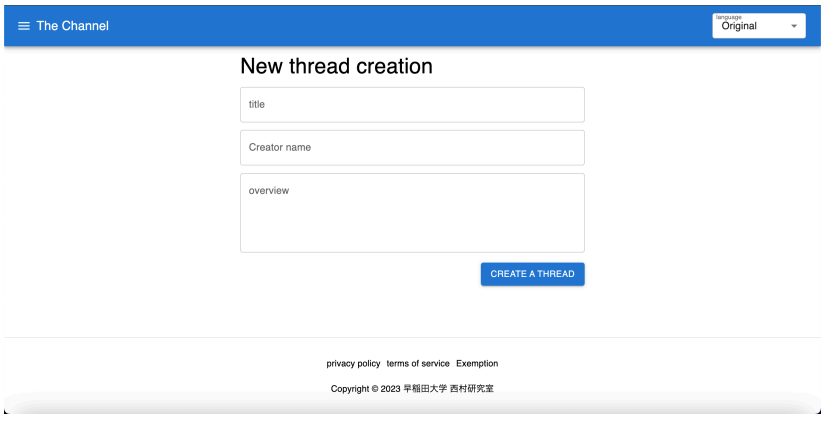


トップページ画面

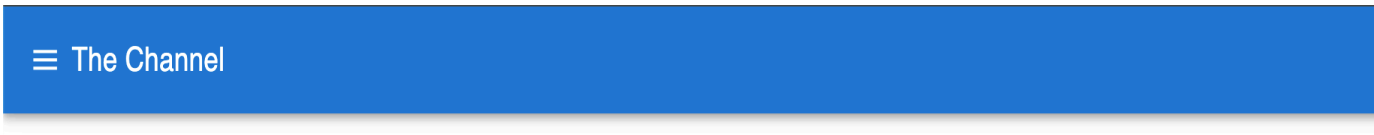
トップページ画面



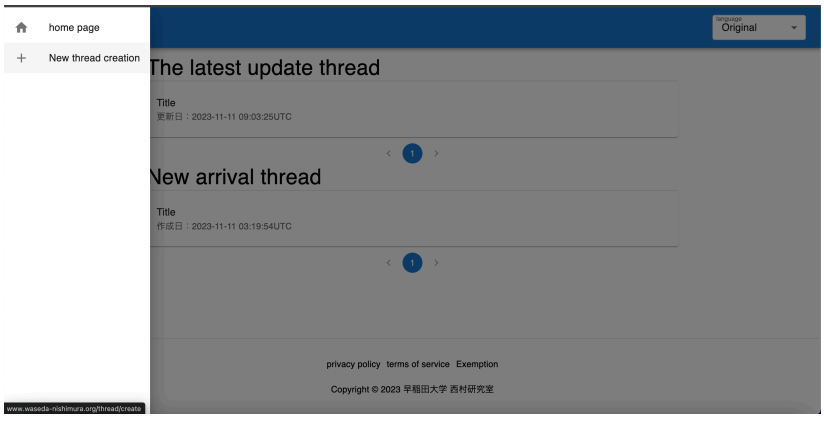
スレッド詳細画面



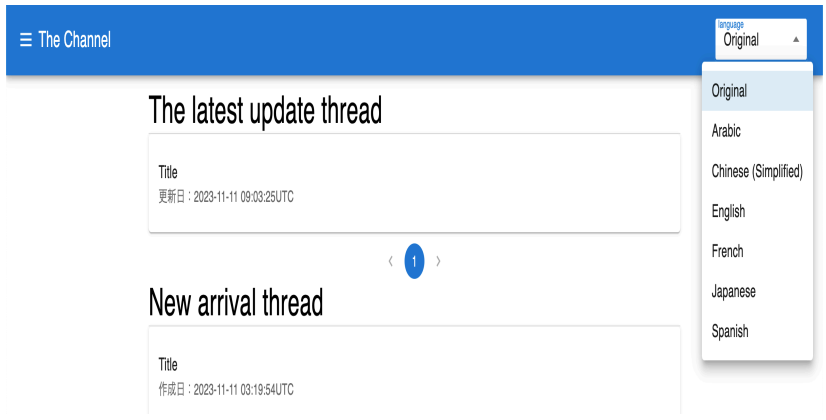
スレッド作成画面



ヘッダ表示



サイドメニュー表示



言語選択表示

サイトの入り口. 表示したいスレッドを選択する画面である. スレッドリストから表示したいスレッドを選択することで，詳細を表示することができる.

スレッド詳細画面スレッドに投稿されたコメントの表示とコメントの投稿を行う画面である. 表示下部のコメント投稿からコメントを投稿することができる.

スレッド作成画面

スレッドを作成する画面である.

ヘッダ表示

常に表示される.The Channel との表示部分をクリックするとトップページ画面に遷移する. 左部にあるメニューボタンをクリックするとサイドメニューが表示される. 右部にある言語選択部分をクリックすると言語選択が表示される.

サイドメニュー表示

トップページボタンをクリックするとトップページ画面へ，スレッド作成ボタンをクリックするとスレッド作成画面へ遷移する. サイドメニュー以外の部分をクリックするとサイドメニューは閉じられる.

言語選択表示

言語を選択するとその言語に翻訳された画面へと遷移する. 言語選択部分以外をクリックすると閉じられる.

フッタ表示

常に表示される. プライバシーポリシー，利用規約，免責事項へのリンクがある. 選択したリンクのページへと遷移する.

以下では，各機能の詳細と画面操作を説明する.

1）スレッド詳細閲覧

スレッド詳細は，トップページ画面に表示されているスレッドリストからスレッドを選択して表示する. スレッドでは，タイトル，概要が上部に表示される．コメントはその下に連なって表示される. コメントはラベル部分とコメント部分に分かれる. ラベル部分には，コメント番号，作成者名，投稿時間が表示される. コメント部分には，投稿されているコメントが表示される．

2）スレッド作成

スレッドを作成することができる. スレッドのタイトル，作成者名，概要を入力する. 全て必須項目であり，空欄があるとその旨を警告するエラーが表示され，スレッドを作成することはできない．

3）コメント投稿

スレッドに対してコメントを投稿することができる. 作成者名とコメントを入力する．コメントとのみ入力必須である. 作成者名が未入力であった場合は，コメントの作成者名には”NO NAME”と表示される．

4）言語選択

閲覧するための言語を選択することができる. ヘッダ右部にある言語セレクター部分をクリックすると，選択することができる言語がドロップダウンメニューとして表示される. 言語をクリックすると画面が更新され，その言語に翻訳された画面に遷移する．ユーザーが選択できる言語は，日本語，英語，中国語，アラビア語，フランス語，スペイン語である. これらは，言語使用者が多い言語である．

5）スレッドリスト

ホーム画面に存在し，直近に更新があったスレッドと作成されたスレッドがそれぞれ 5 件ずつ表示される.5 件以上のスレッドが存在する場合は，リスト下部にあるページャーを選択することで，表示されていないスレッドについても確認することができる．

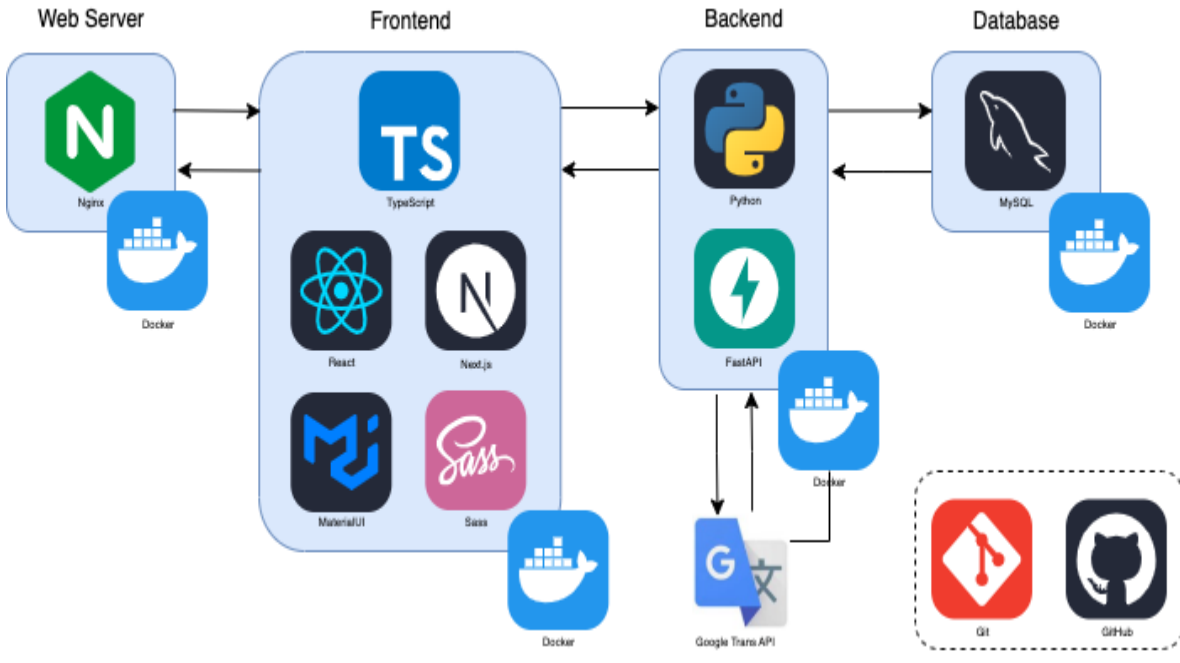
6）ヘッダ

左部にあるメニューボタンをクリックするとサイドメニューを表示することができる.”The Channel”と書かれている部分を選択すると，どの画面からでもトップページ画面に遷移することができる．

2.4　システム構成の概要

本システムは，ユーザーが Web ブラウザを通じて掲示板にアクセスする構成を採用している．

// TODO: アイコンの権利関係確認



技術スタッ

2.4.1 サーバー

本研究では，さくらの VPS をサーバーとして使用している．サーバーのオペレーティングシステムには Ubuntu 22.04.3 LTS を採用した．以下にサーバーの CPU，メモリ，ストレージ，ネットワークインターフェイスに関する詳細な情報を記載する．

メモリ情報

サーバーのメモリは以下の通りである．

- **合計**: 7937 MB
- **使用中**: 1534 MB
- **空き**: 860 MB
- **バッファ/キャッシュ**: 5541 MB
- **利用可能**: 6094 MB
- **スワップ**: 0 MB

ディスク使用状況

サーバーのディスク使用状況は以下の通りである．

- **ファイルシステム**:
  - **tmpfs**: 使用中 794 MB（使用中 1.4 MB，空き 793 MB）
  - **/dev/vda2**: 使用中 788 GB（使用中 40 GB，空き 708 GB）

ディスク構成

サーバーのディスク構成は以下の通りである.

- **ディスク名**: vda
- **サイズ**: 800 GB
- **タイプ**: ディスク
- **パーティション**:
  - **vda1**: 1 MB
  - **vda2**: 800 GB（マウントポイント: /）

CPU 情報

サーバーの CPU に関する情報は以下のとおりである.

- **アーキテクチャ**: x86_64
- **CPU 操作モード**: 32-bit，64-bit
- **アドレスサイズ**: 46 ビット物理，48 ビット仮想
- **バイト順序**: Little Endian
- **CPU 総数**: 6
- **オンライン CPU リスト**: 0-5
- **ベンダー ID**: GenuineIntel
- **モデル名**: Intel Xeon Processor (Cascadelake)
- **CPU ファミリー**: 6
- **モデル**: 85
- **コアあたりのスレッド数**: 1
- **ソケットあたりのコア数**: 1
- **ソケット数**: 6
- **ステッピング**: 5
- **BogoMIPS**: 4199.99
- **サポートされているフラグ**: [フラグのリスト]
- **ハイパーバイザのベンダー**: KVM
- **仮想化タイプ**: 完全仮想化

- **キャッシュの合計**:
  - L1d: 192KiB (6 インスタンス)
  - L1i: 192KiB (6 インスタンス)
  - L2: 24MiB (6 インスタンス)
- **NUMA ノード数**: 1
- **NUMA ノード 0 CPU**: 0-5
- **脆弱性と軽減策**:
  - L1tf: PTE Inversion による軽減
  - Meltdown: PTI による軽減
  - Spectre v1: ユーザーコピー/swapgs バリア，ユーザーポインターのサニタイズによる軽減
  - Spectre v2: IBRS，IBPB 条件付き，STIBP 無効，RSB 充填，PBRSB-eIBRS Not affected による軽減

ネットワークインターフェイスの状態

サーバーのネットワークインターフェイスの設定と状態は以下の通りである.

- **br-f4387cc8d030**:
  - IP アドレス: 172.18.0.1/16
  - MAC アドレス: 02:42:06:18:81:07
  - 受信パケット: 851,033 (2.0 GB)
  - 送信パケット: 893,012 (78.0 MB)
- **docker0**:
  - IP アドレス: 172.17.0.1/16
  - MAC アドレス: 02:42:bf:a4:bf:64
  - 受信パケット: 6,763 (625.9 KB)
  - 送信パケット: 7,761 (43.2 MB)

2.4.2 セキュリティ対策

本研究では，システムのセキュリティ強化のためにファイアウォール（UFW）と AIDE（Advanced Intrusion Detection Environment）を導入した.

ファイアウォール（UFW）の設定

本システムでは，外部からの不正アクセスを防ぎ，内部ネットワークのセキュリティを高めるためにファイアウォール（UFW）を設定している. ファイアウォールは，不正なネットワークトラフィックを検出しブロックすることで，システムを保護する重要なセキュリティ機能である. 以下がその具体的な設定である.

- **状態**: アクティブ
- **ルール**:
  - **ポート 22/tcp**: 全てのアドレスからのアクセスを許可
  - **ポート 443/tcp**: 全てのアドレスからのアクセスを許可
  - IPv6 に関しても同様の設定

このように設定することで，システムは不正なアクセスを効果的に阻止し，同時に必要な通信は許可することができる. 結果として，セキュリティを損なうことなく，システムの柔軟性とアクセシビリティを維持している.

AIDE（Advanced Intrusion Detection Environment）の導入

本システムでは，ファイルシステムの整合性を監視するために AIDE を使用している.AIDE はファイル変更を検出するホストベースの侵入検出システムで，システムに不正な変更が生じた場合に警告を発する.

AIDE の設定は以下の通り.

- AIDE のスケジュール設定：システムは毎日午前 0 時に AIDE を実行し，ファイルシステムの整合性をチェックする.
- メール通知の設定：異常が検出された場合，システムは自動的に指定した E-mail アドレスに通知する.

このセットアップにより，システムの安全性を高め，不正アクセスや変更があった場合に迅速に対応できる.

2.4.3 バックエンド開発

本研究のバックエンド開発においては，Python 言語と FastAPI フレームワークを中心に，多様なライブラリを利用した.Python はバックエンドの主要言語として，FastAPI は主要フレームワークとして採用した. また，翻訳機能は googletrans ライブラリを用

いて実現した.

以下に，本研究において使用したライブラリの一覧を示す.

- anyio: 3.6.2
- cachetools: 5.3.1
- certifi: 2022.12.7
- chardet: 3.0.4
- click: 8.1.3
- fastapi: 0.95.1
- googletrans: 4.0.0rc1
- h11: 0.9.0
- h2: 3.2.0
- hpack: 3.0.0
- hstspreload: 2023.1.1
- httpcore: 0.9.1
- httptools: 0.5.0
- httpx: 0.13.3
- hyperframe: 5.2.0
- idna: 2.10
- mysql: 0.0.3
- mysql-connector-python: 8.0.33
- mysqlclient: 2.1.1
- protobuf: 3.20.3
- pydantic: 1.10.7
- python-dotenv: 1.0.0
- pytz: 2023.3
- PyYAML: 6.0
- rfc3986: 1.5.0
- sniffio: 1.3.0
- starlette: 0.26.1
- typing_extensions: 4.5.0
- uvicorn: 0.22.0
- uvloop: 0.17.0
- watchfiles: 0.19.0
- websockets: 11.0.2

2.4.4 フロントエンド開発

本研究のフロントエンド開発においては，TypeScript 言語と React ライブラリ，Next.js フレームワークを中心に，Material UI を含む多様なライブラリを利用した.TypeScript はフロントエンドの主要言語として，React と Next.js は主要なライブラリとフレームワークとして採用した. また，UI コンポーネントは Material UI を用いて実現した.

依存関係

フロントエンド開発において直接利用されたライブラリの一覧を示す.

- @emotion/react: 11.11.0
- @emotion/styled: 11.11.0
- @mui/icons-material: 5.11.16
- @mui/material: 5.13.2
- @types/gtag.js: 0.0.17
- @types/js-cookie: 3.0.3
- @types/moment: 2.13.0
- js-cookie: 3.0.5
- moment: 2.29.4
- moment-timezone: 0.5.43
- next: 13.3.1
- next-sitemap: 4.2.3
- normalize.css: 8.0.1
- react: 18.2.0
- react-dom: 18.2.0
- react-hook-form: 7.45.2
- sass: 1.62.1

開発用依存関係

開発プロセスを支援するために使用されたライブラリの一覧を示す.

- @types/node: 18.16.0
- @types/react: 18.0.38

- @types/react-dom: 18.0.11
- @typescript-eslint/eslint-plugin: 5.59.0
- @typescript-eslint/parser: 5.59.0
- eslint: 8.39.0
- eslint-config-next: 13.3.1
- eslint-config-prettier: 8.8.0
- eslint-plugin-prettier: 4.2.1
- prettier: 2.8.8
- typescript: 5.0.4

2.4.5 データベースについて

データベースとして MySQL が採用されている. このセクションでは，データベースの構成と，それがシステムにどのように統合されているかについて説明する.

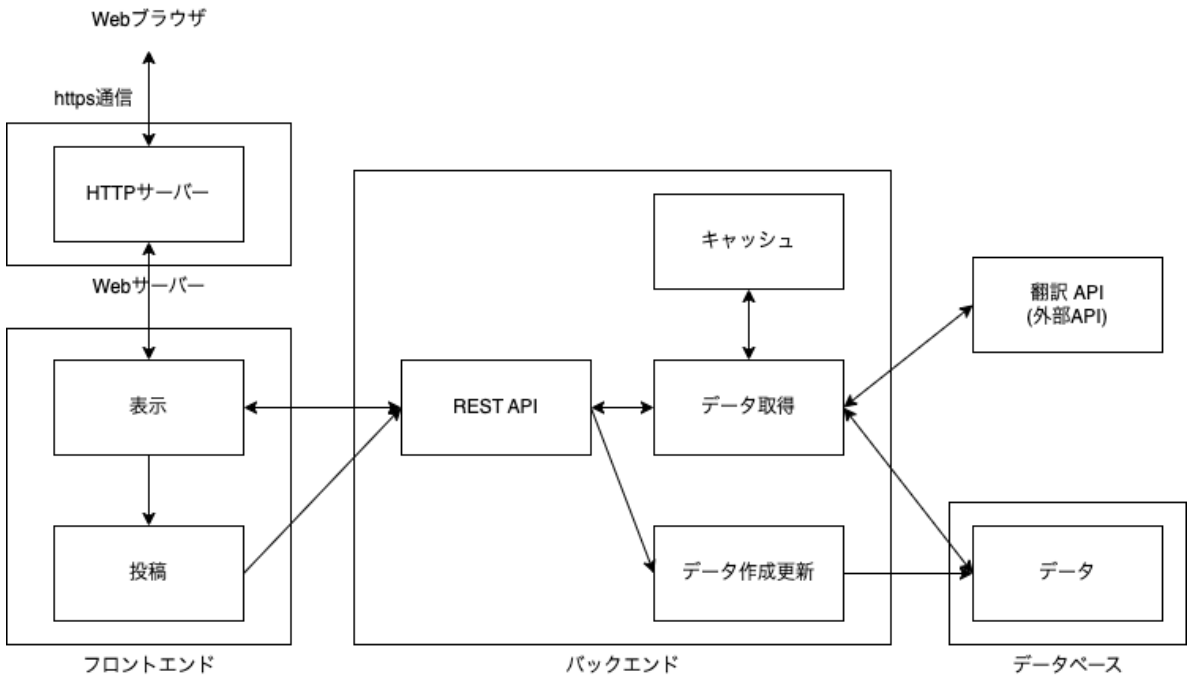
2.4.6 通信の取り扱いと Nginx の役割

本システムにおける通信は https を通じて行われ，セキュリティを確保しつつユーザーのリクエストを効率的に処理する. ユーザーは初めに Nginx を経由してフロントエンドの画面にアクセスし，画面表示に必要なデータを取得する. また，ユーザーからの POST アクセスは Nginx を通じてバックエンドに転送され，処理される. このセクションでは，Nginx がシステム内で果たす重要な役割と，それが通信プロセスにどのように統合されているかについて詳述する.

2.5 コンポーネント設計

この掲示板は，表示コンポーネント，データ取得コンポーネント，データ更新コンポーネントからなる. ユーザー側インターフェイスである Web ブラウザと通信するのは表示コンポーネントである. 表示コンポーネントは，画面を表示するために必要なデータをデータ取得コンポーネントに要求する. データ取得コンポーネントは，その要求を読み取り，適切に加工されたデータを表示コンポーネントに渡す. また，表示コンポーネントは，ユーザーのスレッド作成やコメント投稿に応じてデータ更新コンポーネントを呼び出し，処理を要求する. データ取得コンポーネントはその要求を読み取り，データを適切に加工して保存する.

動的な文章の翻訳はデータ取得コンポーネントで行なっている. 表示コンポーネントが



コンポーネント図

データ取得コンポーネントに処理を要求する際，ユーザーが選択している言語の情報を付属させる. データ取得コンポーネントはデータを取得した後，その言語に翻訳を行い表示コンポーネントにデータを渡す. さらに，システムは翻訳の高速化と効率化のためにキャッシュ機能を実装している. 一度翻訳された文章はキャッシュに保存され，同じ文章の再翻訳が必要な場合，システムはキャッシュから翻訳済みのデータを読み込む. これにより，翻訳のたびに新たに翻訳を行う必要がなくなり，システムの応答時間を大幅に短縮できる. このキャッシュシステムは，特に多言語での交流が活発な掲示板において，ユーザー体験の向上に大きく寄与している.

2.6 コンポーネントの詳細

表示コンポーネント

表示コンポーネントは以下のような構成になっている.

ページはそれぞれ以下のファイルである.

- `/pages/index.tsx` … トップページ
- `/pages/policy.tsx` … プライバシーポリシーページ
- `/pages/terms.tsx` … 利用規約ページ
- `/pages/404.tsx` … 404 エラーページ

- /pages/thread/create.tsx … スレッド作成ページ
- /pages/thread/index.tsx … スレッドページ

// TODO: 表示の流れの図

図は，画面表示の際の表示コンポーネントの振る舞いをあらわしたものである. 表示コンポーネントは，それぞのれページとそのページで使用される部品であるコンポーネントから成り立っている. また，画像やアイコンに関して専用の public フォルダを用意し，そこから読み込んでいる.

データ更新コンポーネント

データ更新コンポーネントは以下のような構成になっている.

- thread_router.py … スレッド関連のルーティングを管理
- thread_application.py … スレッドのアプリケーションロジックを処理
- thread_infrastructure.py … スレッドのインフラストラクチャ関連処理
- comment_router.py … コメント関連のルーティングを管理
- comment_application.py … コメントのアプリケーションロジックを処理
- comment_infrastructure.py … コメントのインフラストラクチャ関連処理
- translation.py … 翻訳機能の実装

// TODO: データ更新の流れの図

上記のファイルリストは，以下のアーキテクチャを採用し作成された. このアーキテクチャは，明確な役割と責任を持つ複数のレイヤーに分かれている.

■ルーターレイヤー

thread_router.py と comment_router.py はルーターレイヤーに属し，外部からのリクエストを適切な処理ロジックに振り分ける役割を担う. このレイヤーは，リクエストの初期解析とルーティングを行い，システムの入口点として機能する.

■アプリケーションロジックレイヤー

thread_application.py と comment_application.py はアプリケーションロジックレイヤーを構成し，スレッドとコメントに関するビジネスロジックを実装する. このレイヤーは，アプリケーションの核心的な機能を担い，データの処理やビジネスルールの適用を行う.

■インフラストラクチャレイヤー

`thread_infrastructure.py` と `comment_infrastructure.py` はインフラストラクチャレイヤーに属し，データベースとのやり取りやデータの永続化を管理する. このレイヤーは，システムのデータ層に関わる処理を担当し，データの整合性と安全性を保証する.

■翻訳機能

`translation.py` は，これらのレイヤーとは別に，システム全体の翻訳機能を提供する. このファイルは，多言語サポートを実現し，異なる言語間でのコミュニケーションを可能にする.

以上のように，各レイヤーは特定の役割を持ち，システム全体の効率的かつ整理された運用を支援する. このレイヤー化されたアーキテクチャにより，システムの拡張性，保守性，およびスケーラビリティが向上している.

2.7 データ設計

Threads テーブル

スレッドの情報を格納するテーブルである.

カラム名	データ型	説明
ThreadID	INT PRIMARY KEY AUTO_INCREMENT	スレッドの ID
Title	VARCHAR(255)	スレッドのタイトル
CreatedAt	TIMESTAMP	作成日時
UpdatedAt	TIMESTAMP	更新日時
UserName	VARCHAR(255)	ユーザー名
Content	LONGTEXT	内容
Language	VARCHAR(8)	言語

Comments テーブル

スレッドに投稿されたコメントを格納するテーブルである.

カラム名	データ型	説明
CommentID	INT PRIMARY KEY AUTO_INCREMENT	コメント ID
ThreadID	INT	スレッド ID
UserName	VARCHAR(255)	ユーザー名
Content	TEXT	コメント内容
CreatedAt	TIMESTAMP	作成日時
Language	VARCHAR(8)	言語

Cookie

Cookie を使用してユーザーの言語選択を記録し，その設定に基づいてユーザーに表示する際に翻訳を行う. この機能により，ユーザーは毎回言語を選択する必要がなく，効率的かつ快適にアプリを利用できる.

Cookie の名称	内容
SelectedLanguage	ユーザーが選択した言語設定

Cookie「SelectedLanguage」は，ユーザーが最後に選択した言語を記録するものである. この情報は次回のウェブサイト訪問時に参照され，ユーザーが以前に選択した言語でインターフェースが表示されるようになる. この機能はユーザビリティを向上させ，多言語対応の効率を高める.

第 3 章

利活用についての分析

本研究で開発された多言語自動翻訳掲示板システムの利活用について分析するためには，ユーザーに使用してもらうことが必要である. このシステムは，異なる文化や言語背景を持つユーザーが交流し，情報を共有するためのプラットフォームとして機能することを目的としている. 従って，多様なユーザーグループを獲得し，継続的に参加を促すことは，システムの成功にとって非常に重要である.

システムの普及に際して，ソーシャルメディアは重要な役割を担う. 本研究では，Twitter を用いてシステムの普及を図った.Twitter は広範なユーザーネットワークを有し，迅速な情報伝播が可能であるため，新しいテクノロジーの告知に適したプラットフォームである. このアプローチにより，ターゲットオーディエンスに直接アピールし，関心を引きつけることが期待される.

参考引用文献

[1] 大向一輝 (2006). SNS の現在と展望-コミュニケーションツールから情報流通の基盤へ-. 情報処理，47(9): 993-1000.

[2] 小川泰弘，福田ムフタル，外山勝彦 (2009). 日本語ーウイグル語翻訳掲示板システム. 言語処理学会 第 15 回年次大会発表論文集，15: 212-215.

[3] 影浦峡 (2017). 改めて，翻訳とは何か：Google NMT が使える時代に. 言語処理学会 第 23 回年次大会発表論文集，23: 931-934.

[4] 亀田倫史 (2022). 機械学習とバイオテクノロジー. 生物工学会誌，100(11): 588.

[5] 中澤敏明 (2017). 機械翻訳の新しいパラダイム：ニューラル機械翻訳の原理. 情報管理，60(5): 299-306.

[6] 野津誠 (2009). 日韓翻訳掲示板「enjoy Korea」終了へ，理由は利用率の低下. 株式会社インプレス, https://internet.watch.impress.co.jp/cda/news/2009/02/12/22405.html（参照日 2023.07.17）

[7] 藤井薫和，重信智宏，吉野孝 (2005). 異文化間コミュニケーションのための機械翻訳を用いたチャットシステム AnnoChat の開発と適用. 情報科学技術フォーラム一般講演論文集，4(3): 437-438.

[8] 船越要，藤代祥之，野村早恵子，石田料亨 (2004). 機械翻訳を用いた協調作業支援ツールへの要求条件ー日中韓馬異文化コラボレーション実験からの知見. 情報処理学会論文誌，45(1): 112-120.

[9] 宮部真衣，吉野孝 (2010). 機械翻訳を介したチャットコミュニケーションにおける精度判定に基づく送信拒否の適用可能性. 情報処理学会論文誌，51(3): 784-795.

[10] 村本麻衣 (2022). 自動翻訳機能からの自立：学習者による気づきを通じて. ドイツ語教育 = Deutschunterricht in Japan / 日本独文学会ドイツ語教育部会 編，26: 119-125.

[11] 吉野孝，藤井薫和，重信智宏 (2006). 異文化間コミュニケーションのためのカスタマイズ可能なユーザインタフェイスを持つチャットシステム CustomChat の開発. 情報処理学会研究報告 = IPSJ SIG technical reports，(60): 13-18.

[12] DeepL (2023). DeepL. https://jobs.deepl.com/（参照日 2023.07.08）

[13] Google Cloud (2023). Translation AI. https://cloud.google.com/

translate?hl=ja（参照日 2023.07.17）
[14] Loki Technology, Inc (2023). 5 ちゃんねる. https://5ch.net/（参照日 2023.07.17）
[15] PyPI (2023a). googletrans 3.0.0. https://pypi.org/project/googletrans/（参照日 2023.07.17）
[16] PyPI (2023b). deepl 1.15.0. https://pypi.org/project/deepl/（参照日 2023.07.17）
[17] Reddit Inc (2023). reddit. https://www.redditinc.com/（参照日 2023.07.08）
[18] Server World (2022). AIDE : ホスト型 IDS. https://www.server-world.info/query?os=Ubuntu_22.04&p=aide（参照日 2023.12.2）