文章要約を用いたタイムスタンプ自動作成システム

芝浦工業大学付属高校	束田繁洸
芝浦工業大学付属高校	佐々木彪雅
芝浦工業大学付属高校	福田啓太
芝浦工業大学	佐々木毅
芝浦工業大学附属中学高等学校	山岡佳代
芝浦工業大学附属中学高等学校	横山浩司

目次

1.はじめに

1.1 開発の背景

近年、オンライン会議が広く普及してる。従来の会議では相手と同じ場所に集合しなければいけなかったが、オンライン会議では相手がどこにいても、手軽にまるで対面で話しているかのように会議を行うことができる。このようなオンライン会議には、後から振り返るために会議の様子を動画として記録できる機能が多く搭載されている。

しかし、動画を振り返る際にどこからがどのような内容かが分かりにくいという問題がある。この問題の一因として、動画にタイムスタンプ（時間情報）がないことが挙げられる。そこで、本稿では会議を記録した動画に対して、「始まる時間：要約」といった形式のタイムスタンプを自動的に作成するためのコンポーネント群を開発した。

1.2タイムスタンプ作成機能の説明

本稿では、動画の特定のセクションの要約と、そのセクションが始まる時間、終わる時間を「始まる時間～終わる時間：要約」という形で示したもので、この情報を基に視聴者が動画の関心のある部分を容易に特定し参照できるようになっているものをタイムスタンプと呼称する。

また本稿では、オンライン会議上で行われる、パワーポイント等を画面共有しながら行うプレゼンテーションを対象として、タイムスタンプ作成機能を開発した。

このシステムは、ウインドウキャプチャ、画面遷移検出、音声検出、cos類似度分析、webソケット通信などの7つの機能から成り立っている。以下の表に各システムの詳細と使用しているコンポーネントを示す。

ウィンドウキャプチャ	window_captchaコンポーネントを使用し、ユーザーがキャプチャしたいウィンドウを指定すると、キャプチャが開始されその時の画像データをscreen_transitionコンポーネントに、録画時間をtimestamp_creationとcos_similarity_analysisコンポーネントに送信する。
画面遷移検出	画面遷移検出コンポーネントを利用しwindow_captchaコンポーネントから送信されてきた画像データを一つ前の画像データと比較し、画面遷移を検出する。検出結果をtimestamp_creationとcos_similarity_analysisコンポーネントに送信する。
システム音声キャプチャ	system_sound_captchaコンポーネントを使用してシステム音声をキャプチャし、文字起こしを行う。出力された文字データはcos_similarity_analysisコンポーネントに送信する。
テキスト要約	cos_similarity_analysisコンポーネントを使用してsystem_sound_captchaコンポーネントから送信されてきた文字データとscreen_transitionコンポーネントから送信されてきた画面遷移のデータをもとに、画面遷移が行われるまでの文章をcos類似度分析とTF-IDF分析を用いて要約を行い、要約された文字データをtimestamp_creationコンポーネントに送信する。

タイムスタンプ作成	timestamp_creationコンポーネントを使用してcos_similarity_analysis コンポーネントからの要約データとscreen_transitionコンポーネントから送信された時間のデータを用いてタイムスタンプを作成する作成された結果はそれぞれwebsocket_to_fastapiコンポーネントとdiscord_botコンポーネントに送られる。
ディスコードボット	discord_botコンポーネントを使用してtimestamp_creationコンポーネントから送られてきた文字データを取得し、discordにメッセージを送る。
http通信	timestamp_creationコンポーネントから受信したタイムスタンプをウェブアプリ上に表示するために使用している。

1.4 開発環境

当コンポーネントの開発環境を下記の表に示す。

OS	windows 10,11
開発環境	visual Studio Code
RTミドルウェア	OpenRTM-aist-2.0.1-RELEASE
Python	Python3.11.5

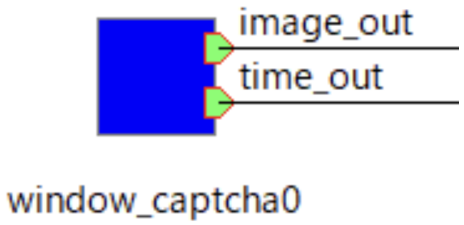
2.コンポーネントの説明

2.1window_captcha（画面キャプチャコンポーネント）

・説明
PyAutoGUIを利用してPCのウィンドウをキャプチャするコンポーネント。ユーザーがキャプチャーしたいウィンドウを指定するとキャプチャが開始された時間を記録し、画像データと録画の時間データを取得する。画像データの取得は0.25秒ごとに行われる。本システムでは取得した画像データはscreen_transitionコンポーネントに、時間データはtimestamp_creationとcos_similarity_analysisに送信する。

・補足説明
time_outからはfloat型で経過時間を秒数で出力する。

・画像



・データポート
以下の表にデータポートの説明を示す。

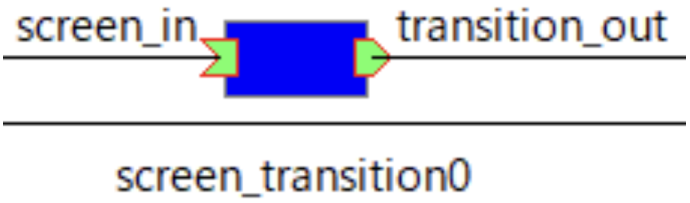
データポート	ポート名	データ型	説明
Outport	image_out	CameraImage	キャプチャされた画像を出力する。
Outport	time_out	TimedFloat	time_outからはfloat型で経過時間を秒数で出力する。

2.2 screen_transition（画面遷移検出コンポーネント）

・説明

window_captchaコンポーネントから送信されてきた画像データを一つ前の画像データと比較し、画面遷移を検出するコンポーネント。送信されてきた画像を10×10のセクションに分け、色相の平均を出し、色相の平均の変化から画面遷移を検出する。画面遷移が行われた場合はTrue、行われていない場合はFalseを文字列で出力する。また本システムでは、timestamp_creationとcos_similarity_analysisコンポーネントに出力結果が送信される。

・画像



・データポート

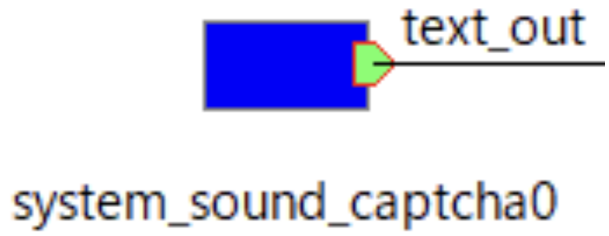
以下の表にデータポートの説明を示す。

データポート	ポート名	データ型	説明
Inport	screen_in	CameraImage	window_captchaコンポーネントから送信されてきた画像データを取得する。
Outport	transition_out	TimedWString	screen_transitionコンポーネントで画面遷移を検出した場合True Falseを文字列で出力する。

2.3 system_sound_captcha（システム音声キャプチャコンポーネント）

・説明
システム音声をキャプチャし、文字起こしを行うコンポーネント。
音声キャプチャの実装には「soundcard」、文字起こしには「whisper」を利用した。音声は5秒毎にキャプチャされ、キャプチャされた音声は文字起こしが行われる。また本システムでは文字データとしてcos_similarity_analysisコンポーネントに送信する。

・画像



・データポート
以下の表にデータポートの説明を示す。

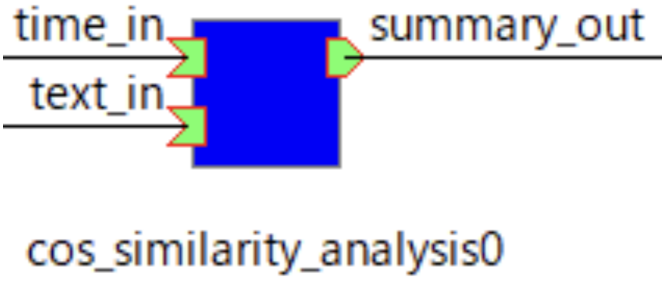
データポート	ポート名	データ型	説明
Outport	text_out	TimedWString	このコンポーネントによって文字起こしされた文字列をcos_similarity_analysisコンポーネントに送信する。

2.4 cos_similarity_analysis（テキスト要約コンポーネント）

・説明
system_sound_captchaコンポーネントから送信されてきた文字起こしのデータとscreen_transitionコンポーネントから送信されてきた画面遷移のデータをもとに、画面遷移が行われるまでの文章をcos類似度分析とTF-IDF分析を用いて要約を行う。

・補足説明
[要約1,要約2]のように、文章の要点をリスト型にまとめて出力する。

・画像



・データポート
以下の表にデータポートの説明を示す。

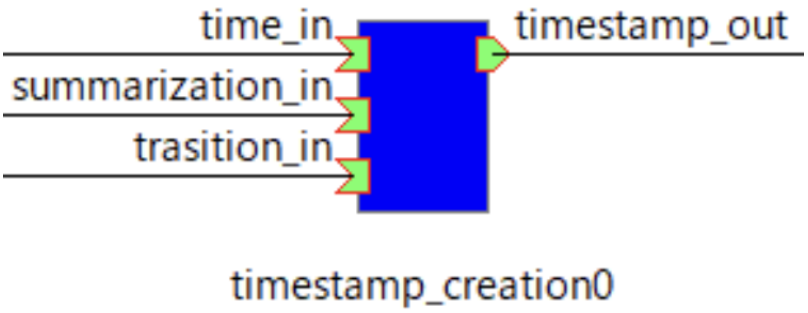
データポート	ポート名	データ型	説明
Inport	transition_in	TimedWString	screen_transitionコンポーネントから送信されてきた画面遷移のデータを取り入れる。
Inport	text_in	TimedWString	system_sound_captchaコンポーネントから送信されてきた文字起こしのデータを取り入れる。
Outport	summary_out	TimedWStringSeq	cos類似度分析とTF-IDF分析を用いて要約を行い、要約された文字データをtimestamp_creationコンポーネントに送信する。

2.5 timestamp_creation(タイムスタンプ作成コンポーネント)

・説明
cos_similarity_analysis コンポーネントから送信されてきた要約データ、screen_transitionコンポーネントから送信されてきたキャプチャが始まってからの時間のデータを用いてタイムスタンプを作成するコンポーネント。Fig.1のようなタイムスタンプが表示される。本システムでは、文字の統合を行ってタイムスタンプとして見やすいように整えられた後、discord_botコンポーネントやwebsocket_to_fastapiに送信する。

・補足説明
timestamp_outからは”0:00~：要約内容”のような形でタイムスタンプが出力される。

・画像



・データポート
以下の表にデータポートの説明を示す。

データポート	ポート名	データ型	説明
Inport	time_in	TimedFloat	window_captchaから送られた録画の時間データを受け取る。
Inport	summarization_in	TimedWStringSeq	cos_similarity_analysisコンポーネントから受け取った要約済みの文章を受け取る。

Inport	trasition_in	TimedWString	screen_transitionコンポーネントから画面遷移が行われたかどうかををtrueかfalseかを文字列として受け取る。
Outport	timestamp_out	TimedWString	出来あがったタイムスタンプをdiscord_botコンポーネントとweb_socket_to_fastapiコンポーネントに送信する。

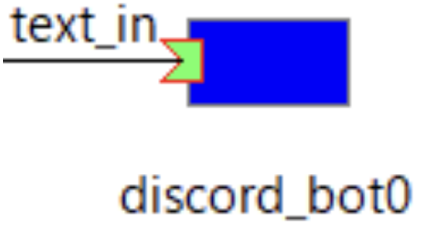
Fig1

2.6 discord_bot（ディスコードボットコンポーネント）

・説明
メッセージングアプリdiscordにデータを表示するためのコンポーネント。timestamp_creationコンポーネントから送信されてきた文字データを取得し、discordにメッセージを送ることができ、ユーザーが日常的に使うメッセージングアプリに結果を表示することで利便性を高めることが期待できる。実装には「request」を利用し、httpリクエストで通信を行い、discordにメッセージを送信している。

・補足説明
当コンポーネントは事前にdiscord deveroperにアクセスしbotを準備する必要がある。マニュアルの「3.2 MeCab,Discord関連の補足説明」をご覧ください。また、後述するDiscordのトークンキー[1]、チャンネルＩＤ[2]をコンフュギュレーション変数に入力してから実行してください。

・画像



・データポート
以下の表にデータポートの説明を示す。

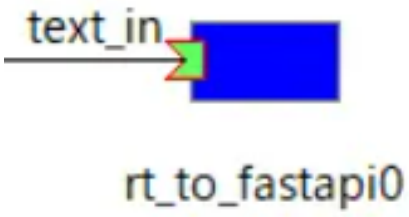
データポート	ポート名	データ型	説明
Inport	text_in	TimedWString	timestamp_creationで出力されたタイムスタンプを受け取る。

2.7 websocket_to_fastapi（FastAPIappコンポーネント）

・説明
Uvicornローカルサーバ、FastAPIを用いたwebsocket通信でブラウザ上にデータを表示するためのコンポーネント。Uvicornを利用したローカルサーバをたて、FastAPIにデータを送受信することによってwebsocket通信を実現している。timestamp_creationコンポーネントから受信したタイムスタンプをウェブアプリ上に表示するために使用されている。

・補足説明
ローカル環境内でサーバを開く場合、同ディレクトリ内のserver.pyでサーバを起動してから実行する必要がある。また、外部のサーバ等でFastAPIapp等を開く場合は、requestで通信するためのURLが必要と　なる。

・画像



・データポート
以下の表にデータポートの説明を示す。

データポート	ポート名	データ型	説明
Inport	text_in	TimedWString	timestamp_creationコンポーネントからのデータを受け取る。

2.本システムの利用手順

3.1 各パッケージ、モデルの説明

FastAPI (fa):
- 音声認識のためのライブラリ。
- インストールコマンド: `pip install fastapi`

Uvicorn (uc):
- 音声認識のためのライブラリ。
- インストールコマンド: `pip install uvicorn`

Pydantic (Pydantic):
- 音声認識のためのライブラリ。
- インストールコマンド: `pip install pydantic`

Requests (requests):
- 音声認識のためのライブラリ。
- インストールコマンド: `pip install requests`

OpenCV (cv2):
- 画像処理やコンピュータビジョン用のライブラリ。
- インストールコマンド: `pip install opencv-python`

PyGetWindow (gw):
- ウィンドウの管理を簡略化するためのライブラリ。
- インストールコマンド: `pip install PyGetWindow`

PyAutoGUI (pyautogui):
- グラフィカルユーザーインターフェース自動化のためのライブラリ。
- インストールコマンド: `pip install pyautogui`

NumPy (np):

- 数値計算用の高性能なライブラリ。
- インストールコマンド: `pip install numpy`

SoundCard (sc):

- サウンドカードへのアクセスを提供するライブラリ。
- インストールコマンド: `pip install soundcard`

SoundFile (sf):

- サウンドファイルの読み書きを行うライブラリ。
- インストールコマンド: `pip install soundfile`

SpeechRecognition (sr):

- 音声認識のためのライブラリ。
- インストールコマンド: `pip install SpeechRecognition`

SpeechRecognition (sr):

- 音声認識のためのライブラリ。
- インストールコマンド: `pip install SpeechRecognition`

一括インストールコマンド

```
pip install PyGetWindow pyautogui soundcard SoundFile
SpeechRecognition fastapi uvicorn pydantic requests
```

3.2 MeCab,Discord関連の補足説明

「MeCab」の利用手順

日本語の形態素解析を行うためのライブラリ。MeCabを使用するには、以下の手順が必要となる（Windows環境を想定）

①Mecabのウェブページを開く https://taku910.github.io/mecab/#download

②Binary package for MS-Windowsよりmecab-0.996.exeをダウンロード

③ダウンロードしたexeファイルを実行する※辞書の文字コードを選択する画面ではUTF-8を選択

④環境変数「PATH」に「MeCabをインストールしたフォルダ/bin」を追加

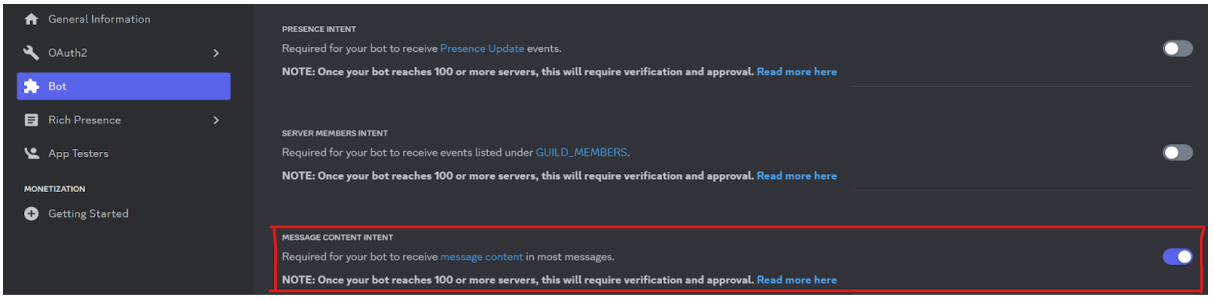
「Discordコンポーネント」の事前準備

discord.pyを動かす前にはまず、discord deveroper にアクセスをしbot側を準備する必要がある。

①discord deveroperにアクセス https://discord.com/developers/docs/intro

②New applicationからbotを作成

③botタブのMessage content intentをチェックする



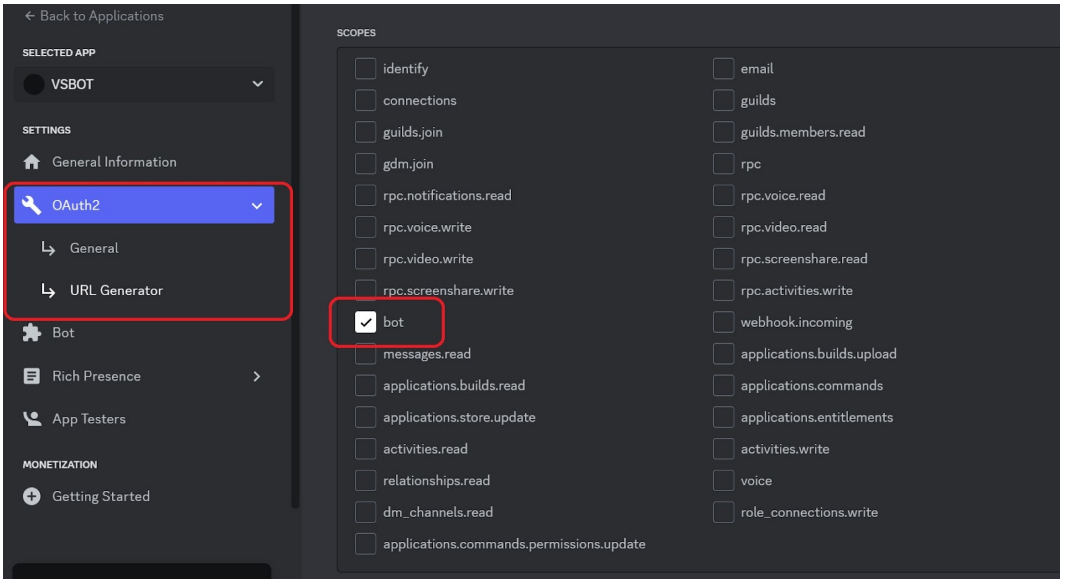
・この項目はbotがメッセージの内容を受け取るか否かの項目である。

④出来上がったbotのトークンキーの取得

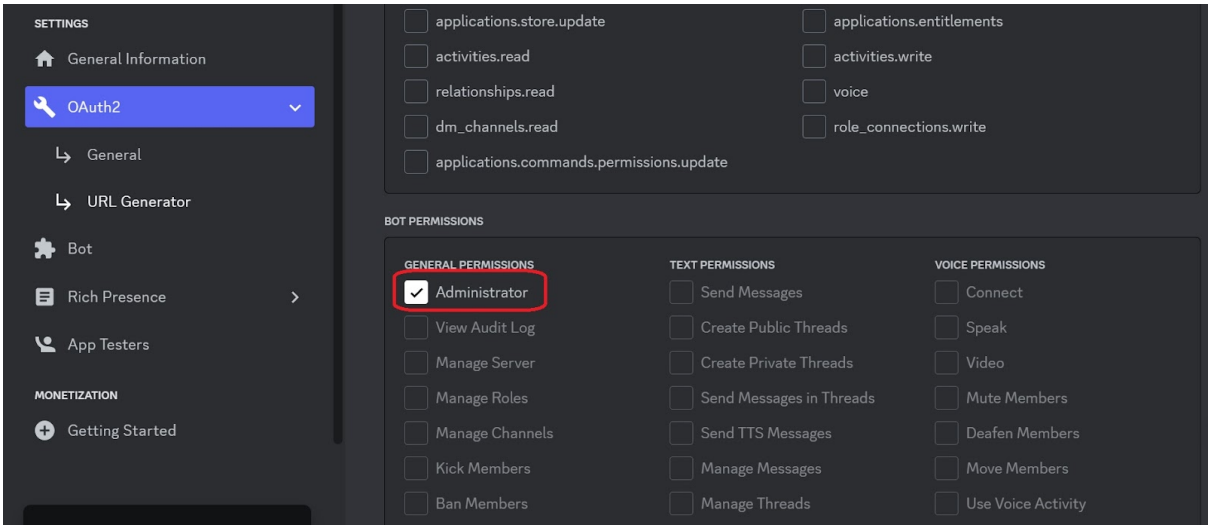
・ここで出したトークンキーは後に使用する。

⑤実際に使用するDiscordサーバーに招待

・setteing、OAuth2のURLgeneratorからscorp欄のbotにチェックする。



・また、チェック後にすぐ下に出現するBot PremissionsのAdministratorにチェックすると生成されるURLへアクセス後に完了となる。



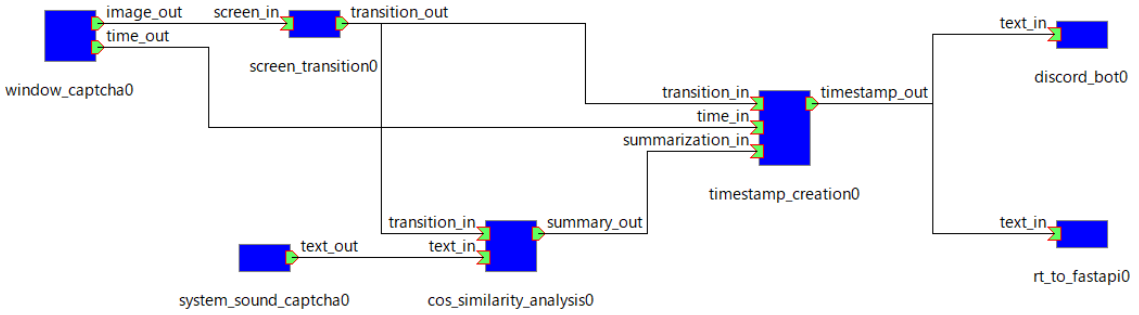
⑥チャンネルIDの取得

以下の画像のようにbotを招待したDiscordサーバーの利用したいテキストチャンネルIDを取得する。

以上のことよりdiscordコンポーネントが使用可能になる。

3.3 コンポーネントの接続

①ダウンロードしたコンポーネント群を以下のように接続

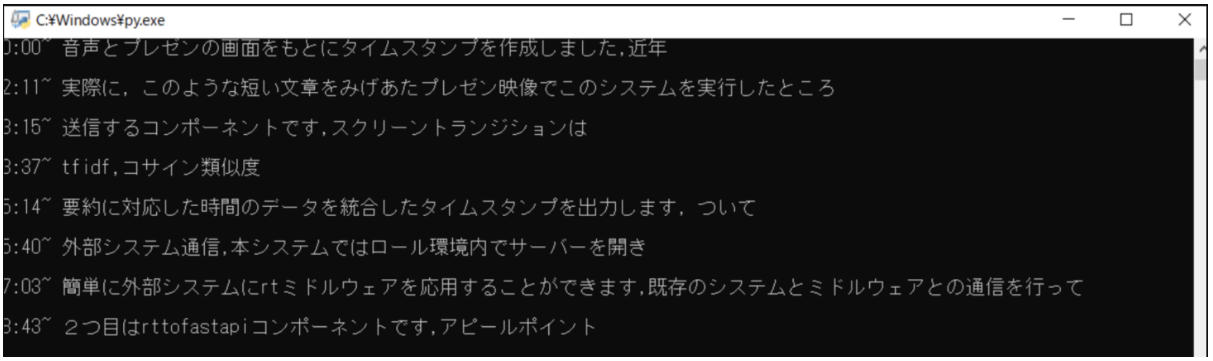


②実行する前にdiscord_botコンポーネントに先ほど記述したDisocrdのトークンキー[1]チャンネルID[2]をコンフィギュレーション変数に入力

③すべてのコンポーネントを実行し、キャプチャしたいウィンドウを指定

3.4 実際の実行画面

以下の画像は本システムの実行結果です。



4.参考文献

https://atmarkit.itmedia.co.jp/ait/articles/2102/05/news027.html
https://taku910.github.io/mecab/#download
https://discord.com/developers/docs/intro