Japanese Text Processing System

A Multi-Agent OCR, NLP, and LLM-Powered Visual Understanding Platform

---

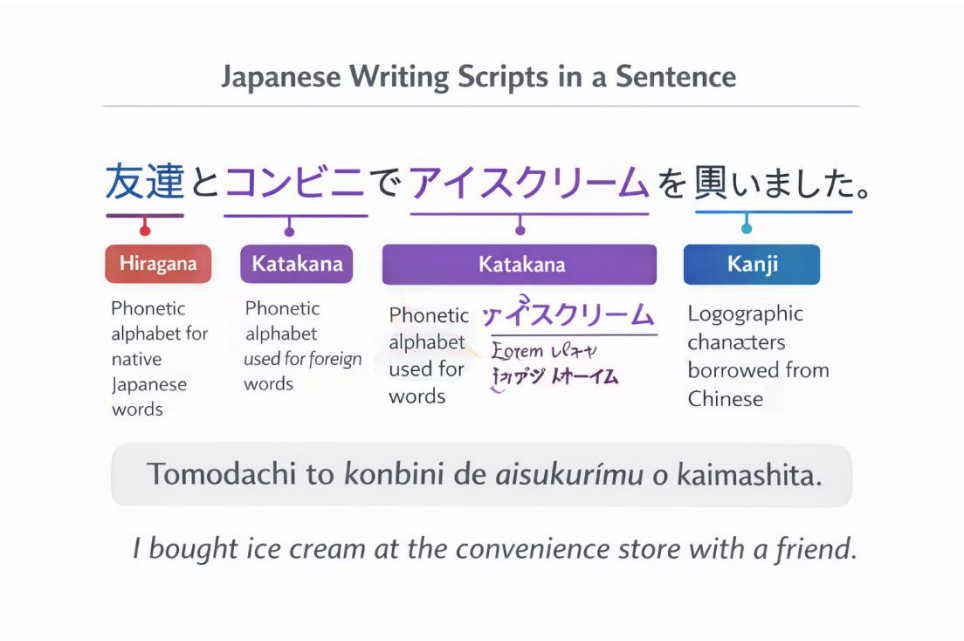
Introduction

The Japanese Text Processing System is a multi-agent platform designed to extract, understand, and visually annotate Japanese text found in images. Japanese text processing presents unique linguistic and technical challenges due to the absence of whitespace, the coexistence of multiple writing systems (Kanji, Hiragana, and Katakana), and the heavy reliance on contextual grammar. This system addresses these challenges by combining classical OCR techniques, modern natural language processing, dictionary-based lexical analysis, and large language model reasoning within a modular and extensible architecture.

The primary goal of the system is to support **Japanese language learners** by:

- Extracting text accurately from images
- Explaining vocabulary and grammar
- Providing natural translations
- Visually annotating kanji with furigana readings

The system is specifically designed for language learners, educational tools, assistive reading technologies, and research applications that require not only accurate text extraction, but also meaningful linguistic explanations and visual feedback.



By Shivam Kumar

High-Level Architecture

At a high level, the system operates as a FastAPI service that accepts image uploads and processes them through a LangGraph-managed workflow. The workflow coordinates four primary agents: the OCR Agent, the NLP Agent, the LLM Agent, and the Visualization Agent. Each agent contributes specific information to a shared processing state, which is progressively enriched as it flows through the system.

By using LangGraph as the orchestration layer, the system avoids hidden control flow and instead relies on an explicit state machine. This makes execution predictable, debuggable, and easy to extend with additional processing branches in the future.

The processing pipeline consists of four main agents:

1. **OCR Agent** – Extracts Japanese text and layout metadata from images
2. **NLP Agent** – Tokenizes text and enriches it with readings and meanings
3. **LLM Agent** – Produces translation and grammar explanations
4. **Visualization Agent** – Adds furigana annotations directly onto the image

These agents are coordinated through a **state-driven workflow**, where each stage consumes and enriches a shared processing state.

OCR Agent: Optical Character Recognition

The OCR Agent is responsible for extracting raw Japanese text from input images. It uses Tesseract OCR with Japanese language models enabled. The agent does not simply return plain text; instead, it extracts detailed spatial metadata for each detected text element, including bounding box coordinates and confidence scores.

This spatial information is critical for downstream processing, particularly for visual annotation. The OCR Agent also groups individual text elements into logical lines based on vertical alignment, preserving reading order. Low-confidence OCR results are filtered out to reduce noise and improve the quality of subsequent linguistic analysis.

Processing Steps

1. Loads the image and converts it into a NumPy array
2. Applies Tesseract OCR configured for Japanese (jpn)
3. Extracts individual text elements with bounding boxes and confidence scores
4. Filters low-confidence noise while retaining complex kanji

5. Groups text elements into logical lines using vertical proximity
6. Produces a clean, concatenated text string for downstream NLP analysis

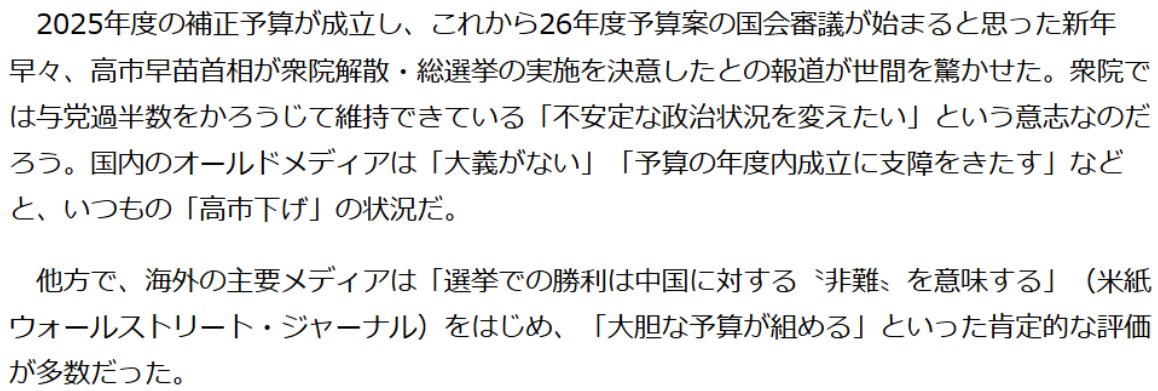
Output

The OCR Agent outputs:

- The original image array
- A list of text elements with bounding box coordinates
- Line-grouped text data
- The complete extracted text as a single string

The OCR Agent operates independently of language understanding. Its sole responsibility is to convert pixels into structured textual data, making it easy to replace with a cloud-based OCR solution in the future if higher accuracy or multilingual support is required.

example image :



2025年度の補正予算が成立し、これから26年度予算案の国会審議が始まると思った新年早々、高市早苗首相が衆院解散・総選挙の実施を決意したとの報道が世間を驚かせた。衆院では与党過半数をかろうじて維持できている「不安定な政治状況を変えたい」という意志なのだろう。国内のオールドメディアは「大義がない」「予算の年度内成立に支障をきたす」などと、いつもの「高市下げ」の状況だ。

他方で、海外の主要メディアは「選挙での勝利は中国に対する〝非難〟を意味する」（米紙ウォールストリート・ジャーナル）をはじめ、「大胆な予算が組める」といった肯定的な評価が多数だった。

NLP Agent: Japanese Linguistic Processing

The NLP Agent handles the linguistic complexity of Japanese text. Once raw text is extracted, this agent performs morphological analysis using the nagisa tokenizer, which segments Japanese text into words and assigns part-of-speech tags. Because Japanese does not use spaces, this step is essential for meaningful downstream analysis.

After tokenization, the NLP Agent converts each token into Hiragana, Katakana, and Romaji using pykakasi. This provides phonetic readings that are especially valuable for language learners. The agent also detects whether a token contains Kanji characters and, if so, attempts to retrieve dictionary meanings using JMdict via the jamdict library.

Tokenization and POS Tagging

The agent uses **Nagisa**, a Japanese morphological analyzer, to:

- Segment text into words
- Assign part-of-speech tags
- Handle Japanese text without explicit word boundaries

Reading Generation

Using **pykakasi**, each token is converted into:

- Hiragana
- Katakana
- Romaji

Dictionary Integration

When available, **JMdict (via Jamdict)** is used to retrieve English meanings for kanji-containing tokens. Meanings are selectively attached to avoid clutter and improve clarity.

Output

The NLP Agent produces:

- A list of enriched tokens
- A vocabulary dictionary mapping kanji words to meanings
- Metadata indicating kanji presence and linguistic role

example output :

```
"vocabulary": [
  {
    "kanji": "年",
    "hiragana": "ねん",
    "meaning": "year"
  },
  {
    "kanji": "度",
    "hiragana": "ど",
    "meaning": "time (three times, each time, etc.); times"
  },
  {
    "kanji": "補正",
    "hiragana": "ほせい",
    "meaning": "correction; revision"
  },
  {
    "kanji": "予算",
    "hiragana": "よさん",
    "meaning": "estimate (of costs); budget"
  },
  {
    "kanji": "成立",
    "hiragana": "せいりつ",
    "meaning": "formation; establishment"
  },
```

The NLP Agent enriches each token with linguistic metadata while remaining agnostic to grammar interpretation or semantic reasoning. Its output forms the lexical backbone of the system.

LLM Agent: Semantic Understanding and Grammar Analysis

The LLM Agent is responsible for high-level semantic understanding of the Japanese text. It uses the Groq LLM API to generate a natural English translation of the full text and to identify and explain important grammar patterns present in the input.

Rather than performing word-by-word translation, the LLM Agent focuses on producing learner-friendly explanations. It is guided by carefully constructed prompts that request structured output, including a full translation and a list of grammar patterns with explanations. The agent includes robust response parsing logic, as well as fallback mechanisms to ensure usable output even when the LLM response deviates from the expected format.

Analysis Workflow

1. The full extracted Japanese text is embedded into a structured prompt
2. The LLM is instructed to act as an **expert Japanese teacher**
3. The model is asked to:
    - Translate the entire text naturally into English
    - Identify and explain important grammar patterns
4. The response is parsed into structured fields

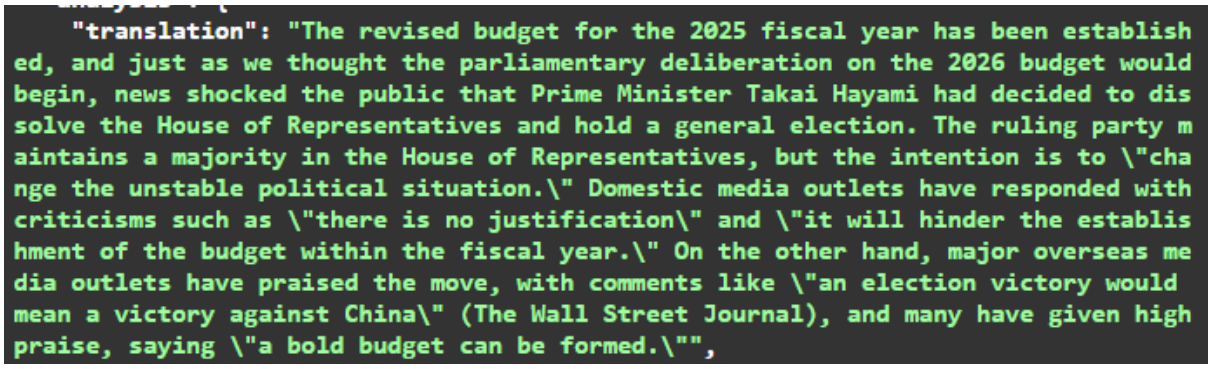
The prompt is designed to encourage **educational depth**, not just literal translation.

Output Structure

The LLM Agent returns an LLMAnalysis object containing:

- A natural English translation of the text
- A list of grammar patterns with explanations

example output :



```
    "grammar_patterns": [
      "*の**: This pattern is used to indicate possession or relationship between two nouns. For example, \"2025 年 度 の 補正 予算\" (the revised budget for the 2025 fiscal year), where \"の\" indicates that the budget belongs to or is related to the 2025 fiscal year.\n2. **と**: This pattern is used to quote or indicate thoughts and speech. For example, \"「不安定 な 政治 状況 を 変え た い」 と いう 意志\" (the intention to \"change the unstable political situation\"), where \"と\" indicates that the phrase is a quote or a thought.\n3. **だろう**: This is a polite and somewhat formal way of expressing speculation or assumption. For example, \"「不安定 な 政治 状況 を 変え た い」 と いう 意志 な の だ ろう\" (the intention to \"change the unstable political situation\" is likely), where \"だろう\" is used to express a polite assumption.\n4. **をはじめ**: This pattern is used to indicate the beginning of a list or a series of items. For example, \"「選挙 で の 勝利 は 中 国 に 対する を 意味 する」（米 紙 ウォ ー ル スト リー ト ・ ジ ヤー ナル）を は じ め\" (starting with \"an election victory would mean a victory against China\" (The Wall Street Journal)), where \"をはじめ\" indicates that this is the first item in a list of examples.\n5. **だつた**: This is the past tense of the verb \"だす,\" which is used to express the existence or appearance of something. For example, \"いつ た 定 的 な 評価 が 多数 だ つた\" (there were many positive evaluations), where \"だつた\" indicates that the evaluations existed or appeared in the past.\n6. **て**: This pattern is used to indicate a cause-and-effect relationship or a sequence of action
```


This agent encapsulates all large language model interactions, isolating external dependencies and making it straightforward to switch models or providers in the future.

Visualization Agent: Educational Image Annotation

The Visualization Agent transforms linguistic understanding into visual learning aids. Using the original image and OCR bounding box information, this agent overlays furigana (Hiragana readings) above Kanji characters.

Annotation Strategy

- Furigana is placed **above kanji characters**
- Text is horizontally centered relative to the kanji
- Vertical placement adapts dynamically based on detected line spacing
- Semi-transparent backgrounds ensure readability
- Font size adjusts automatically to avoid overlap

Font Handling

The agent supports multiple Japanese-compatible fonts across platforms:

- Windows (Meiryo, Yu Gothic, MS Gothic)

- macOS (Hiragino)
- Linux (Noto Sans CJK)

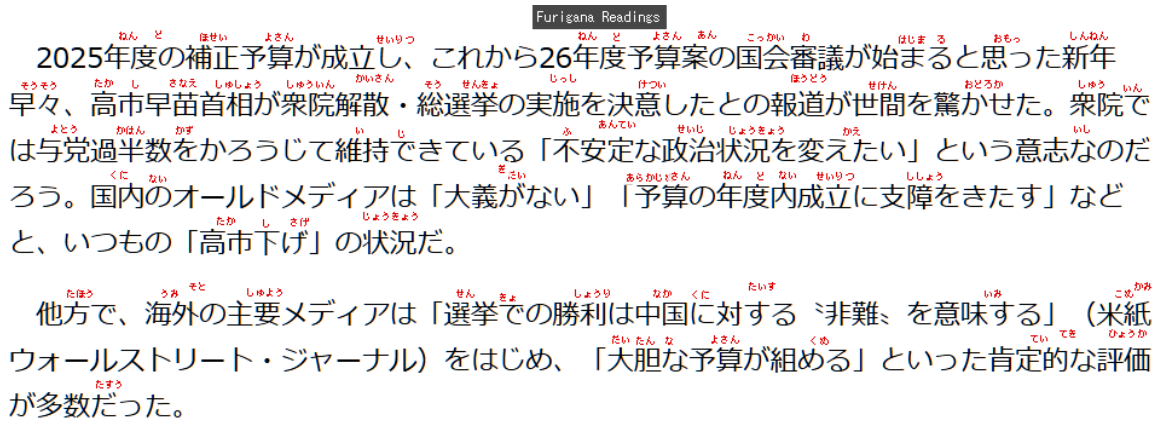
If no suitable font is found, a fallback font is used with a warning.

Output

The Visualization Agent produces:

- A visually annotated image with furigana
- Robust fallback behavior if rendering fails
- Compatibility with OpenCV for saving and further processing

example output :



The Visualization Agent produces a final annotated image that visually connects written Japanese text with its pronunciation, significantly enhancing comprehension for learners.

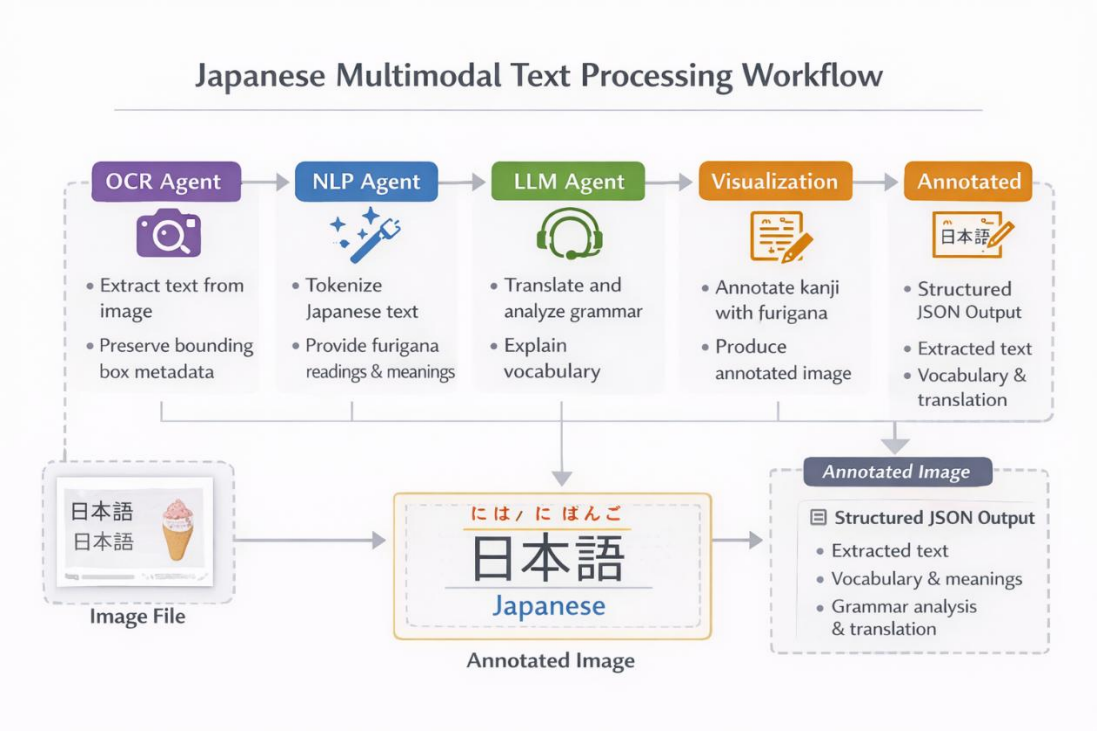
LangGraph Workflow Orchestration

The entire system is orchestrated using LangGraph, which manages execution as a state machine. The workflow begins with OCR processing and proceeds sequentially through NLP analysis, LLM interpretation, and visualization. Each agent receives and updates a shared ProcessingState object, ensuring transparent data flow.

Workflow Nodes

1. **OCR Node** – Extracts text and layout metadata
2. **NLP Node** – Tokenizes and builds vocabulary

3. **LLM Node** – Produces translation and grammar analysis
4. **Visualization Node** – Generates annotated image



LangGraph provides a clean abstraction for agent orchestration, avoiding tightly coupled function calls and enabling future extensions such as conditional branches, retries, or parallel processing stages.

FastAPI Application Layer

The FastAPI layer exposes the system as a RESTful API. It handles file uploads, input validation, error handling, and response formatting. The /process endpoint accepts an image file and returns structured JSON output along with a reference to the annotated image file.

Endpoints

- **GET /**
  Returns system metadata, agent descriptions, and supported features
- **POST /process**
  Accepts an image file and returns:
  - Extracted text statistics
  - Vocabulary list
  - Grammar analysis and translation

- Annotated image path
- Processing metrics

FastAPI's automatic OpenAPI documentation makes the service easy to explore and integrate with external applications. The API layer is stateless, allowing horizontal scaling and deployment in containerized environments.

Conclusion

This Japanese Text Processing System demonstrates a approach to multimodal language understanding. By combining OCR, NLP, dictionary resources, large language models, and visual annotation within a carefully orchestrated multi-agent architecture, the system delivers both technical robustness and educational value.

The modular design, explicit agent roles, and LangGraph orchestration make the system easy to extend, maintain, and adapt to future requirements. Whether used as a learning aid, a research tool, or a foundation for more advanced language applications, this project provides a solid and professional implementation of Japanese text understanding from images.

By Shivam Kumar