# Analysis of the Ichiran Library

**Author**: MiniMax Agent **Date**: 2025-06-22

## 1. Introduction

This report provides a comprehensive analysis of the ichiran library, a collection of linguistic tools for the Japanese language. It is important to note that despite the user’s initial request referring to a “go-ichiran” library, the actual library found is written in **Common Lisp**, not Go. This analysis will, therefore, focus on the Common Lisp library and provide guidance on how to build a web API around it.

## 2. Library Capabilities and Main Features

The ichiran library offers a range of features for Japanese language processing:

* **Text Segmentation:** It can break down Japanese sentences into individual words and particles.
* **Romanization:** It provides romanized (romaji) versions of Japanese text.
* **Dictionary Integration:** It integrates with the JMdictDB dictionary database to provide detailed information about Japanese words, including definitions, readings, and parts of speech.
* **Kanji Analysis:** It can provide information about individual kanji characters, leveraging the Kanjidic2 dictionary.
* **Command-Line Interface:** It includes a command-line interface (CLI) for direct interaction.
* **Web Interface:** A web interface for the library is also available at [ichi.moe](https://ichi.moe).

## 3. API Usage Patterns and Examples

The primary way to interact with the ichiran library is through its Common Lisp functions. Here are some examples of how to use the library’s API:

### 3.1. Romanization and Word Segmentation

The romanize function can be used to get the romanized version of a Japanese sentence, along with detailed information about each word.

**Example in Common Lisp:**

(ichiran:romanize "一覧は最高だぞ" :with-info t)

**Output:**

"ichiran wa saikō da zo"  
(("ichiran" . "一覧 【いちらん】\n1. [n,vs] look; glance; sight; inspection\n2. [n] summary; list; table; catalog; catalogue")  
 ("wa" . "は\n1. [prt] 《pronounced わ in modern Japanese》 indicates sentence topic\n2. [prt] indicates contrast with another option (stated or unstated)\n3. [prt] adds emphasis")  
 ("saikō" . "最高 【さいこう】\n1. [adj-no,adj-na,n] best; supreme; wonderful; finest\n2. [n,adj-na,adj-no] highest; maximum; most; uppermost; supreme")  
 ("da" . "だ\n1. [cop,cop-da] 《plain copula》 be; is\n2. [aux-v] 《た after certain verb forms; indicates past or completed action》 did; (have) done\n3. [aux-v] 《indicates light imperative》 please; do")  
 ("zo" . "ぞ\n1. [prt] 《used at sentence end》 adds force or indicates command"))

### 3.2. Building a Web API

While ichiran is a Common Lisp library, you can still build a web API around it. One common approach is to create a web server in a language like Python (using a framework like Flask or FastAPI) and have it call the ichiran CLI or a small Lisp process to perform the analysis. The server would then format the output as JSON and expose it through an API endpoint.

**Example API server in Python (using Flask):**

from flask import Flask, request, jsonify  
import subprocess  
  
app = Flask(\_\_name\_\_)  
  
@app.route('/analyze', methods=['POST'])  
def analyze\_text():  
 text = request.json.get('text')  
 if not text:  
 return jsonify({'error': 'No text provided'}), 400  
  
 # Call the ichiran CLI  
 process = subprocess.run(['ichiran-cli', '-i', text], capture\_output=True, text=True)  
 if process.returncode != 0:  
 return jsonify({'error': 'Error analyzing text'}), 500  
  
 # In a real implementation, you would parse the output of the CLI  
 # and return a structured JSON response.  
 return jsonify({'result': process.stdout})  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run(debug=True)

## 4. Installation Requirements and Dependencies

The ichiran library has the following dependencies:

* **SBCL (Steel Bank Common Lisp):** The recommended Common Lisp implementation.
* **Quicklisp:** A library manager for Common Lisp.
* **PostgreSQL:** The database used to store the dictionary data.
* **JMDict and Kanjidic2:** The dictionary files.

The installation process involves:

1. Setting up the database and importing the dictionary data.
2. Configuring the settings.lisp file with the correct database connection parameters.
3. Loading the library and its dependencies using Quicklisp.

A Dockerized version is also available, which simplifies the installation process.

## 5. Japanese Dictionary and Kanji Analysis Functionalities

The ichiran library provides robust dictionary and kanji analysis capabilities through its integration with JMdictDB and Kanjidic2.

* **JMdictDB:** This provides comprehensive information about Japanese words, including multiple definitions, readings (hiragana/katakana), and parts of speech.
* **Kanjidic2:** This provides detailed information about individual kanji characters, including their meanings, readings (on’yomi and kun’yomi), and radical information.

The library’s romanize function, as shown in the example above, demonstrates how this information can be retrieved for each word in a sentence.

## 6. Limitations and Considerations

* **Language Barrier:** The library is written in Common Lisp, which may be a barrier for developers who are not familiar with the language.
* **Performance:** The performance of the library, especially the initial loading and database initialization, can be slow. For real-time applications, it is recommended to keep the Lisp process running and communicate with it via a web server or other means.
* **Experimental Features:** The segmentation and romanization algorithms are described as “experimental,” which means they may not be perfect and could have limitations.

## 7. Conclusion

The ichiran library is a powerful tool for Japanese language processing, but it is important to be aware that it is a Common Lisp library, not a Go library. Despite this, it can be effectively used to build a web API for Japanese dictionary and kanji analysis. By creating a wrapper around the ichiran CLI or by communicating with a persistent Lisp process, developers can leverage its rich features to create a robust and accurate Japanese language API.