Keeping LLMs from Being Distracted: Grade-Aware Kanji Reading Estimation Fully Executable inWeb Browsers for Japanese Education

Yo Ehara

Proceedings of the 39th Pacific Asia Conference on
Language, Information and Computation (PACLIC 39)

Emmanuele Chersoni, Jong-Bok Kim (eds.)

2025

Keeping LLMs from Being Distracted: Grade-Aware Kanji Reading Estimation Fully Executable in Web Browsers for Japanese Education

Yo Ehara
Tokyo Gakugei University / 4-1-1 Nukuikita-machi, Koganei-shi, Tokyo, Japan.
ehara@u-gakugei.ac.jp

Abstract

Automatic reading (ruby or furigana) generation of kanji is a fundamental task in Japanese natural language processing (NLP), particularly for educational purposes. Several web services already provide grade-aware annotations, assigning readings only to characters not included in each elementary school grade list; however, these services usually rely on a server. They upload pupil texts, process remotely, and return with annotations. Such a design has practical problems: server overload caused by simultaneous access to many tablets and security risks due to transmission of text that may contain personal information. We introduce a browser-based system that places the reading estimation model and grade filter inside the JavaScript engine of each client. After a single-page load, all processing occurs locally; therefore, no further network communication is required. In experiments, the proposed system achieved an accuracy comparable to existing server-side approaches while removing the server load entirely. Large language models such as GPT-5, are becoming increasingly popular in education, yet they still make reading estimation errors, even on short examples. When GPT-5 is asked to simultaneously solve elementary math problems and assign kanji readings, its math accuracy degrades significantly, indicating that LLMs are heavily distracted. These results suggest that our method not only offers a practical solution for latency and privacy issues in obtaining high-quality results but also prevents LLMs from being distracted.

1 Introduction

Automatic reading (ruby or furigana) estimation of kanji characters is a fundamental Japanese NLP task with educational applications. Several web services now provide grade-sensitive ruby, adding furigana only to characters that exceed the set of kanji that is officially taught in each grade of an elementary school. However, most of these services transmit user text to a server where readings are inferred and returned. This architecture raises three practical concerns: (i) server bottlenecks when many pupils simultaneously access the system on tablets, (ii) information security risks for texts containing personal data, and (iii) reduced availability under unstable network conditions, such as during natural disasters.

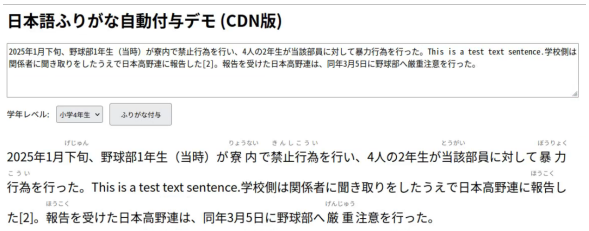

Figure 1: Our System. Once the page that uses JavaScript is loaded, the Japanese reading estimation of the input text is conducted fully on browsers without communicating with servers. When a user specifies the reading of a kanji character, the text with the reading attached appears below the text box. In this example, readings are attached to kanji characters for 4th graders and above. The translation of this example is available in later sections.

We propose a client-side system that performs both reading estimation and grade-sensitive ruby annotation entirely within the browser's JavaScript engine, eliminating the need to upload the input text. After the initial page load, all processing occurs locally, thereby ensuring privacy, resilience, and scalability (Figure 1).

In Japan, the individual kanji characters taught in each elementary school grade are specified by the government. The process of attaching readings to kanji characters at or above the specified grade level is an important process in Japanese education; however, because it is deterministic, the large language model (LLM) often makes mistakes or becomes distracted. Our evaluation measures both

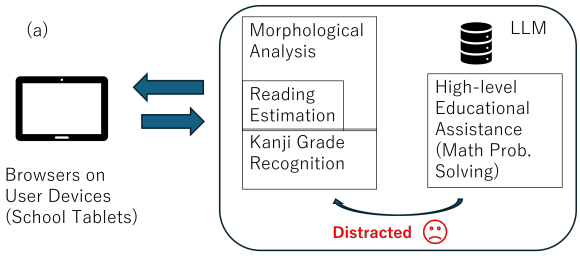

Figure 2: A previous approach combined with LLM educational assistance. LLMs are distracted in assigning grade-aware readings to kanji, which degrades the quality of educational assistance.

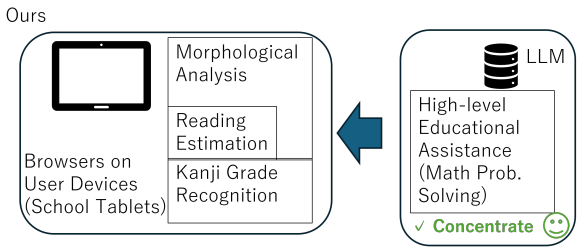

Figure 3: Our approach combined with LLM educational assistance. LLMs concentrate on educational assistance because grade-aware kanji reading assignment is conducted on browsers of the user device (school tablets).

the reading-estimation accuracy and precision of the grade-level filter that determines which of the kanji receive ruby. Because the target characters are predetermined by official gradewise kanji lists, near-perfect accuracy can be achieved in principle. LLMs such as ChatGPT, however, sometimes hallucinate in grade assignments, leading to lower accuracy. Therefore, our experiments highlight the complementarity of traditional rule-based or statistical NLP running in the browser with LLM-based approaches.

This study is the first to create a grade-specific reading estimator for Japanese that works solely on a browser. Additionally, it demonstrates that technical collaboration is necessary when utilizing LLMs for educational purposes, such as entrusting deterministic processing of LLMs to the browser side, in addition to improving LLM performance.

The proposed method is expected to be effective, especially in educational support using LLM. In a recent approach, the LLM performs all the complex tasks (Figure 2). However, with this method, the LLM is distracted by grade-aware kanji reading estimation, which reduces the quality of the essential educational support.

In contrast, in our approach, the task of grade-aware kanji reading estimation, a task that traditional natural language processing is good at, does not require GPUs, but only a browser running on a user device such as a school tablet (Figure 3). The recent tablet browsers are sufficiently powerful to perform morphological analysis, reading estimation, and kanji-level recognition for the amount of text that a user is supposed to read. This allows the LLM to focus on more essential educational support. In this study, we experimentally demonstrated the effectiveness of our approach by considering a specific educational support setting.

The resources such as the demo system can be found at `https://rebrand.ly/okuranukana`.

Translation of Figure 1 The example was taken from a recent Wikipedia article, hence has no copyright issues. It is translated as follows: “In late January 2025, a first-year baseball team member (at the time) engaged in prohibited behavior within the dormitory, and four second-year students committed acts of violence against the said team member. The school conducted interviews with the parties involved and reported the incident to the Japan High School Baseball Federation. Upon receiving the report, the Japan High School Baseball Federation issued a severe reprimand to the baseball team on March 5 of the same year.”

We selected this news because it is domestic to Japan and, as it was a hot topic at the time of submission in Japanese education, elementary school students should be able to read this news.

2 Related Work

Japanese reading estimation underpins tasks, such as text-to-speech synthesis, educational support, and accessibility enhancement (Yasuda et al., 2018; Nagata, 1999; Hoshino and Morise, 2021). Early systems relied on server-side morphological analyzers, such as MeCab and JUMAN++ (Kudo et al., 2004; Kawahara and Kurohashi, 2013), the high accuracy of which come at the cost of network latency and privacy risks when deployed in web applications. Client-side alternatives have emerged with kuromoji.js (Asano, 2014), one of the few analyzers implemented entirely in JavaScript. However, its reading-estimation accuracy has not been rigorously benchmarked against standard corpora.

Methodologically, reading estimation has progressed from statistical to neural approaches. Conditional random fields achieved strong performance when integrated into morphological analyz-

ers (Kudo et al., 2004), and subsequent studies extended segmentation and furigana generation to educational texts (Nagata, 1999). Neural models now exploit contextual embeddings to predict readings with improved prosody for text-to-speech (TTS) (Yasuda et al., 2018) and joint grapheme–phoneme conversion (Hoshino and Morise, 2021). Despite these advances, most neural systems assume a server-side execution, leaving a gap for fully in-browser solutions.

Evaluation typically follows text-input research, consisting of character error rate, first-candidate accuracy, and word error rate, to quantify phonetic conversion errors unified under the framework of Soukoreff and MacKenzie (Soukoreff and MacKenzie, 2003; Morris et al., 2004; Wang and Woodland, 2003). Recent progress in WebAssembly (WASM) and libraries such as TensorFlowjs has reduced the performance gap between browser and native runtimes (Team, 2018; Gardner et al., 2018), thereby enabling complex natural language processing (NLP) models to run locally. The MeCab-WASM and similar ports illustrate this trend; however, a systematic assessment of their reading-estimation quality remains limited. To address this deficiency, the present study offers the first comprehensive benchmark of kuromoji.js on Kyoto University Web Document Leads Corpus (KWDLC) and Advanced Japanese IME Evaluation (AJIMEE)-Benchmark, comparing it with MeCab-WASM, thereby clarifying the extent to which browser-based Japanese NLP has matured.

This study is also connected to research on the interplay between vocabulary and readability. Although not focused on Japanese language education, detailed investigations into the relationship between vocabulary knowledge and readability in English as a second language can be found in (Ehara, 2023, 2022, 2021, 2019, 2018; Ehara et al., 2012).

3 Proposed System

Figure 1 shows our system. The proposal system comprises two components. The first is reading estimation. Japanese, like Chinese, is a language that does not divide words into syllables, requiring that word boundaries be automatically determined. Only Chinese characters require reading estimation, as hiragana and katakana are phonetic characters and do not require reading estimation. In Japanese processing, the system divides Japanese text into words and estimates their readings and parts of speech, conventionally called morphological analysis.

The proposed system requires a morphological analyzer that runs in a browser. Two methods have recently been developed for this purpose. One uses a morphological analyzer written entirely in JavaScript. The other uses Web Assembly, a mechanism that allows virtual machine code to run in a browser, and outputs a morphological analyzer written in C or another language as a Web Assembly binary. We used kuromoji.js to perform a morphological analysis of Japanese text, as it is designed based on JavaScript and has a wealth of usage examples. This is the JavaScript version of Kuromoji, a Java-based open-source Japanese morphological analyzer [1]. We used kuromoji.js version 0.1.2 in combination with the lexicon in the dictionary developed by the Information-Technology Promotion Agency of Japan (IPADIC) (Lab, 2007). We ran the engine under Node.js v18.17.0 and verified identical behavior in chromium-based browsers, ensuring that our results can be generalized to typical client-side deployment.

`kuromoji.js` tokenize Japanese text and estimate readings on the fly. Next, we need an educational kanji database that enumerates characters officially taught in Grades 1–6. For Japanese elementary schools, a clear list of kanji characters taught by grade level is provided by the Japanese government. This list consists of 1,026 characters. These characters are directly loaded into a dictionary-type variable in JavaScript. The surface form and reading of each morpheme are then cross-referenced with the specified grade level in the kanji database; characters that exceed the learner's scope are flagged. Finally, the system injects `<ruby>` elements for the flagged items and streams the fully annotated text back to the DOM, enabling seamless display without network latency or privacy concerns. All user interaction and visualisation are handled by a minimalist HTML/CSS/JavaScript frontend, ensuring platform independence and eliminating the need for server-side processing.

3.1 Implementation Example

The following is an example of furigana annotation for 2nd-grade elementary students:

`Input:` 私は毎日学校に通っています。
`Output:` 私[わ た し]は 毎 日[ま い に ち]学校[がっこう]に通[かよ]っています。

[1] https://www.atilika.com/en/kuromoji/

Here, furigana is not added to kanji learned in 1st grade such as “学,” “校,” and “日.”

4 Evaluation Datasets and Methods

4.1 Evaluation Datasets

We conducted experiments using three corpora that differed in register, genre, and target applications. **KWDLC** (Kyoto University Web Document Leads Corpus; (University, 2011)) supplies 150 web sentences comprising 2,383 morphemes and 6,120 characters, each annotated with morpheme boundaries, part-of-speech tags, and ruby information. **AJIMEE-Bench** (azooKey Project, 2023), built on the Japanese Wikipedia Input Error Dataset v2 (Tanaka et al., 2021), contains 200 kana–kanji pairs, half presented with preceding context and half without, distributed under the CC-BY-SA 3.0 license. Finally, the **Educational Kanji Dataset** was constructed based on the Ministry of Education’ s official grade-level kanji list (Ministry of Education, Culture, Sports, Science and Technology (MEXT), 2017), which covers 1,026 characters across Grades 1–6 (80, 160, 200, 202, 193, and 191 characters, respectively) and serves as the gold standard for grade-aware furigana annotation.

These corpora jointly enable a balanced assessment of general reading estimation accuracy, robustness to input errors, and pedagogical suitability for primary education.

As stated in Section 3, the proposed system is based on kuromoji.js.

MeCab-WASM provides a WebAssembly port for MeCab and functions as a comparative system. Compiled from MeCab 0.996 and linked to the same IPADIC dictionary, this module is executed entirely within WebAssembly-compliant browsers. The shared lexicon and browser-native runtime isolate the differences for the tokenizer, allowing a fair comparison of speed and accuracy with JavaScript-native `kuromoji.js`.

4.2 Evaluation Metrics

Considering the characteristics of reading estimation and kana-kanji conversion, we employed the following metrics:

4.2.1 Character-level Evaluation

$$\text{CER} = \frac{\text{Levenshtein Distance}(r, h)}{|r|} \tag{1}$$

where r is reference reading; h is hypothesis reading; and $|r|$ is reference reading length. Then, Character Accuracy is defined as $(\text{Correct Characters}/\text{Total Characters}) \times 100\%$.

4.2.2 Morpheme-level Evaluation

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

$$\text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

where TP is true positive (correct); FP is false positive (over-detection); and FN is false negative (underdetection).

4.2.3 Perfect Match Evaluation

$$\text{Accuracy@1} = \frac{\text{Perfect Match Count}}{\text{Total Items}} \times 100\% \tag{5}$$

5 Experimental Results

5.1 KWDLC Evaluation Results

We conducted a detailed analysis of 150 sentences containing 2,383 morphemes using KWDLC.

5.1.1 Overall Performance Metrics

Table 1 presents the overall results of the KWDLC evaluation.

5.1.2 Performance Analysis by Sentence Length

Table 2 shows performance according to sentence length category.

Performance degradation is observed with increasing sentence length. This is primarily attributed to increased vocabulary diversity in longer sentences, which makes morpheme boundary identification more difficult.

5.1.3 Error Factor Analysis

Table 3 shows the breakdown of error factors.

The largest error factor was the morpheme boundary mismatch (51.3%), whereas pure reading estimation errors were only 1.3%. This suggests that the reading-estimation function of kuromoji.js is highly accurate.

5.2 AJIMEE-Bench Evaluation Results

The AJIMEE-Bench evaluation by Kuromoji.js demonstrated excellent performance.

Metric	Value	Description
Character Reading Accuracy	**83.17%**	5,090 chars/6,120 chars
Morpheme Perfect Match	**47.38%**	1,129 morphemes/2,383 morphemes
Morpheme Boundary Accuracy	48.72%	(1,129+32)/2,383
Precision	45.16%	1,129/2,500
Recall	47.38%	1,129/2,383
F1 Score	**46.24%**	Harmonic mean

Table 1: Overall Performance Metrics in KWDLC Evaluation

Category	Sentences	Precision	Recall	F1 Score
Medium (10-20 chars)	18	59.3%	64.5%	61.8%
Long (20-30 chars)	59	57.2%	59.3%	58.3%
Very Long (30+ chars)	73	42.0%	43.2%	42.6%

Table 2: Performance by Sentence Length Category (KWDLC Evaluation)

5.2.1 Overall Performance Metrics

Table 4 presents the results of the AJIMEE-Bench evaluation.

5.2.2 CER Distribution Analysis

Table 5 provides detailed CER distributions.

Notably, 89% of the items achieved a CER below 10%, without large errors exceeding 30%.

5.2.3 Context Dependency Analysis

Table 6 shows the performance differences for with and without context.

Interestingly, a higher accuracy was achieved without context. This indicates that kuromoji.js employs dictionary-based methods without considering context, which leads to errors in items requiring context-dependent reading disambiguation.

5.3 Inter-system Comparison

Table 7 presents a performance comparison between the evaluation systems.

As for "*", the value was calculated from the KWDLC evaluation character accuracy of 83.17% (100-83.17=16.83%).

5.4 Performance Evaluation

Table 8 shows the evaluation results of the grade-level furigana annotation system.

In higher grades, appropriate annotation rates improved. This is attributed to the clarification of furigana annotation targets with an increase in the number of kanji learned.

6 Discussion

Reading estimation and kana–kanji conversion require fundamentally different evaluation strategies that diverge in direction, focus, and contextual dependence. Kana–kanji conversion is assessed for the ability to transform purely phonetic input into kanji-kana mixed text, which demands high precision in selecting the correct homophonic alternative among many candidates. In contrast, reading estimation proceeds in the opposite direction, converting mixed text into a phonetic script, and therefore hinges on resolving homographs rather than homophones. Because each kanji character typically maps to a limited set of readings, dictionary information alone can often disambiguate the output, whereas kana-kanji conversion must rely heavily on broader sentential context (e.g. distinguishing 「公園」 *park* from 「講演」 *lecture*). These contrasting requirements necessitate distinct metrics and test suites, underscoring the fact that the performance of an algorithm cannot be extrapolated from one task to another without careful validation.

A methodological mismatch implies that a single algorithm can yield discordant results when evaluated under two paradigms. Empirical evidence from our experiments shows that systems optimized for kana-kanji conversion do not automatically excel in reading estimation and vice versa. Consequently, comparative studies must explicitly frame their evaluation protocols or risk drawing misleading conclusions about model capabilities across tasks that although superficially related, embody different linguistic challenges.

The kuromoji.js node offers a practical browser-

Error Factor	Proportion	Description
Morpheme Boundary Mismatch	51.3%	Different morpheme segmentation
Reading Estimation Error	1.3%	Correct boundary, wrong reading
Perfect Match	47.4%	Both boundary and reading correct

Table 3: Breakdown of Error Factors (KWDLC Evaluation)

Evaluation Metric	Value	Remarks
Average CER	**2.14%**	Extremely low error rate
Accuracy@1	**81.50%**	High perfect match rate
Perfect Match Items	163/200	81.5% perfect reading estimation

Table 4: Overall Performance Metrics in AJIMEE-Bench Evaluation

CER Range	Items	Proportion	Cumulative
0% (Perfect match)	163	81.5%	81.5%
0-10%	15	7.5%	89.0%
10-20%	15	7.5%	96.5%
20-30%	7	3.5%	100.0%
30%+	0	0.0%	100.0%

Table 5: Detailed CER Distribution (AJIMEE-Bench Evaluation)

Category	Items	Average CER	Difference
Without Context	100	1.44%	Baseline
With Context	100	2.84%	+1.40%

Table 6: Context Dependency Analysis (AJIMEE-Bench Evaluation)

native solution that balances accuracy, robustness, and deployment simplicity. On the AJIMEE-Bench dataset, the engine achieved a character error rate of 2.14% and an accuracy @1 of 81.50% while maintaining stable performance without catastrophic errors. Its JavaScript implementation eliminates external API dependencies, enabling high-speed client-side processing that preserves user privacy and ensures availability, even under constrained network conditions. These strengths position `kuromoji.js` as an attractive baseline for educational applications that require lightweight yet reliable reading estimation.

6.1 Limitations Analysis

1. **Lack of context consideration**: Degraded accuracy in context-dependent reading disambiguation

2. **Morpheme boundary differences**: Evaluation difficulties across different corpora

3. **New words and proper nouns**: Challenges in processing words not in dictionary

6.2 Educational Application Potential

Implementation and evaluation of the grade-level furigana annotation system yielded the following insights:

1. **Practical accuracy**: Achieved over 90% appropriate annotation rate

2. **Scalability**: Light operation in browser environment

3. **Customizability**: Flexible configuration based on Course of Study guidelines

These results indicate that kuromoji.js can serve as a practical solution in educational support.

7 LLM Evaluation

7.1 Accuracy of LLM-based reading estimation

We conducted an analysis on how well LLMs can estimate readings by considering Japanese elementary grades. In this short example of Figure 1, the readings are assigned to kanji characters that fourth graders can read, which means that fifth grade and above, can assign the readings to kanji. ChatGPT's GPT-5 assigns excessive readings to six instances. Claude 4.1 and GPT-o3 only assigned excessive reading to one instance. In cases where readings were assigned, all the models assigned accurate readings.

7.2 Measuring LLM Distraction by Multi-tasking

Some may wonder whether it would be better to have LLM perform reading estimations simultaneously. However, studies have shown that when LLMs are tasked with multiple assignments, they become distracted and make errors (Xu et al., 2024).

System	Dataset	CER	Accuracy@1	Remarks
kuromoji.js	AJIMEE-Bench	2.14%	81.50%	This evaluation
kuromoji.js	KWDLC character-level	16.83%*	-	*Estimated value
MeCab-WASM	AJIMEE-Bench	-	-	Under evaluation

Table 7: Inter-system Performance Comparison

Grade Level	Test Sentences	Appropriate Rate	Over-annotation	Under-annotation
Grade 1	50	92.3%	4.1%	3.6%
Grade 3	50	94.7%	2.8%	2.5%
Grade 6	50	96.2%	1.9%	1.9%

Table 8: Grade-level Furigana Annotation System Evaluation Results

Metric	**Count**	**Percentage**
Total Ground Truth Grade-aware Reading Assignments	106	–
Total System Assigned	50	–
Exact Matches	8	–
Mismatches	2	–
Missing Annotations	69	–
Extra Annotations	40	–
Recall	–	7.55%
Precision	–	16.00%
F1 Score	–	10.26%

Table 9: GPT-5 Grade-aware Reading Assignment Predictive Performance

Therefore, if the LLM is tasked with solving problems while performing reading estimation for elementary school students, errors are likely to occur.

To measure the extent to which the LLM is distracted, we conducted the following experiment: The experiment used elementary mathematics from massive multitask language understanding (MMLU) (Hendrycks et al., 2020), which is widely used to evaluate LLM performance. These are simple problems at the elementary school level. MMLU is an English dataset; however, a Japanese version called JMMLU [2] has been published, which includes native Japanese speakers who have confirmed that the problems are valid in Japanese.

Using JMMLU, we had the LLM solve Japanese-translated elementary mathematics problems while estimating the problem text. In our experiments, we used GPT-5, which was released on August 7, 2025. Because GPT-5 automatically switches models during operation, we selected the GPT-5 automatic mode and input the problem statements. The experiment was conducted using an account subscribed to ChatGPT Pro. First, we selected questions 1 through 30 from the elementary-mathematics section of MMLU and provided the following instructions in Japanese: "For the following question, please output the problem statement with furigana for all kanji characters that are at or above the fourth grade level, and then select the correct answer from the four options provided and write your answer immediately after the problem statement. The options are labeled A, B, C, and D for the first, second, third, and fourth options, respectively. “ This was in Japanese, and the results were collected. This is referred to as the ” w/distraction case."

Next, the chat from the ChatGPT account was deleted to prevent it from being recorded. Then, the first instruction in the problem statement was replaced with “please output the problem statement as-is,” and the problems were solved again. The results were then collected. This is referred to as the “w/o distraction case.” In the w/o distraction case, only one out of 30 questions was incorrect, whereas in the w/ distraction case, seven sentences were incorrect. Even with simple elementary school math problems, there is significant distraction, making it impractical for the LLM to perform reading and inference simultaneously. This demonstrates the effectiveness of the proposed method, which performs reading inferences on the browser.

We also measured the reading-assignment accuracy of GPT-5 in Table 9. The results indicated significant challenges in the system's ruby annotation capability, with a recall of only 7.55%, suggesting that most of the required ruby annotations were not generated. A precision of 16.00% indicated that even among the annotations produced, accuracy was limited.

[2] https://github.com/nlp-waseda/JMMLU

7.3 Local LLM Results

We experimented to see what the results would be using a smaller local LLM model. We conducted the experiment using **gpt-oss-20b** [3], which was released by OpenAI on August 5, 2025. The experiment was conducted using the same procedure as that for ChatGPT's GPT-5. The results showed that while the model made 76 reading estimations, only five were correct, resulting in an accuracy rate of 4.72%.

Furthermore, the accuracy rate for elementary school math problems was low, with only 14 out of 30 questions answered correctly, four unanswered, and an overall accuracy of 46%. However, when the reading estimation task was not assigned, there were no unanswered questions, and 18 of the 30 questions were answered correctly, with an accuracy rate of 60%.

These results indicate that the issue of distraction caused by reading estimation remains severe even with a local model, making it infeasible to have LLM perform reading estimation simultaneously while having elementary school students read the text.

Our evaluation demonstrated that the GPT-5 distracted model exhibits significant limitations in Japanese ruby annotation tasks, with an F1 score of only 10.26%. The primary failure mode was low recall (7.55%), indicating that the model failed to identify most of the kanji requiring annotation. These results underscore the continued importance of specialized NLP tools for structured linguistic tasks and suggest that current general-purpose language models are not suitable replacements for dedicated ruby annotation systems in education or accessibility applications.

8 Conclusion

This study presents a comprehensive evaluation of Japanese reading estimation systems that run entirely in JavaScript, with particular focus on the widely used `kuromoji.js`. Our experiments show that the library delivers a performance suitable for real-world deployment, achieving a character-error rate of 2.14% and *Accuracy@1* of 81.50%. In examining the evaluation protocol, we highlight crucial methodological distinctions between conventional kana and kanji conversion benchmarks and the reading estimation task, thereby providing clearer criteria for future assessments. The prototype classroom application further demonstrated that current browser-based Japanese text-processing technologies have matured to a practical stage.

Beyond technical validation, the results indicate four broad areas of impact. First, in education, accurate on-the-fly reading estimation enables automatic furigana annotation and individualized reading support tools. Second, in terms of accessibility, the offline operation of the system offers robust reading assistance to visually impaired or low-connectivity users without risking data leakage. Third, for learners of Japanese as a second language, the same technology can underpin multilingual-level adaptive learning aids. Finally, because all the processing occurs locally, this approach offers privacy benefits that are increasingly demanded by modern web applications.

This study conducted experiments using a single zero-shot prompt, assuming practical use in real educational settings. While from a scientific research perspective, techniques such as Chain-of-Thought, role-based, or sequential prompting could potentially yield higher performance even with LLMs alone, applying such prompting strategies in actual classroom environments remains difficult in practice.

For elementary school students in Japanese, adding furigana is an essential feature in NLP using LLMs, but even LLMs such as GPT-5, which have received significant investment, have been shown to make mistakes. As shown in several studies, such as (Xu et al., 2024), LLMs struggle to perform multiple tasks simultaneously. Therefore, furigana assignments should be performed in a browser, whereas LLM should focus on processing educational content as a desirable approach for future NLP applications in education. In particular, this study demonstrated that simultaneously performing reading inference with an LLM for Japanese elementary school math problems resulted in a seven-fold increase in the error rate of the LLM. As research on the use of LLMs for educational purposes has increased, it has become impractical to have LLMs perform multiple tasks simultaneously. Our results suggest the usefulness of methods such as ours that adapt individually to learners on the browser side.

8.1 Future Work

In this study, we focused on Japanese language processing; however, because it is common for Chi-

[3] https://huggingface.co/openai/gpt-oss-20b

nese characters to have multiple readings, the same mechanism can be applied to Chinese characters. Specifically, after using a Chinese word segmenter that runs on a browser, the Chinese readings of kanji characters can be estimated. In general, since there are fewer variations in Chinese character readings than in Japanese, where each character has two types of readings (an ancient Chinese-derived reading and a Japanese-specific reading), we expect that the word segmenter will operate as smoothly as in this study, even on a browser.

In Japan, the Global Innovation Gateway for All (GIGA) School Project is being implemented to provide elementary school students with tablet computers through government budget subsidies. Similar initiatives are being implemented in regions such as Taiwan and Hong Kong, and efforts to promote tablet use are ongoing in Mainland China. This study provides a useful method for efficiently obtaining effective educational support from LLMs using tablets in elementary schools.

Limitations

This paper's estimation is limited to certain language models. As language models are evolving quickly, other language models may be able to estimate the readings of Japanese texts more correctly or may be able to conduct complicated educational assistance with being less distracted by estimating the readings of Japanese texts.

We did not test calling multiple LLM tasks simultaneously for each task. While this may solve the problem of the performance drop caused by multiple task following, this does not solve the problems caused by client-server network communications.

Claude Code was used to support our coding in experiments. ChatGPT was used for automatic English proofreading.

Ethical Considerations

Since we conducted our experiments using only publicly-available datasets which allows research purpose use, we believe that our study has no ethical issues. We prioritized measuring the performance of kanji reading estimation using the proposed approach and conducting computer experiments to show that the proposed method improves the quality of LLM educational support by eliminating distractions from reading estimation. Therefore, user studies have not been conducted at this stage, and hence this paper does not involve ethical issues related to user studies.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Number 22K12287 and by JST, PRESTO Grant Number JPMJPR2363. We are deeply grateful to the anonymous reviewers for their constructive feedback.

References

Takuya Asano. 2014. kuromoji.js: Japanese morphological analyzer in javascript. https://github.com/takuyaa/kuromoji.js. JavaScript implementation of Japanese morphological analyzer.

azooKey Project. 2023. Ajimee-bench: Advanced japanese ime evaluation benchmark. https://github.com/azooKey/AJIMEE-Bench. GitHub repository.

Yo Ehara. 2018. Building an English vocabulary knowledge dataset of Japanese English-as-a-second-language learners using crowdsourcing. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Yo Ehara. 2019. Uncertainty-aware personalized readability assessments for second language learners. In *18th IEEE International Conference On Machine Learning And Applications, ICMLA 2019, Boca Raton, FL, USA, December 16-19, 2019*, pages 1909–1916. IEEE.

Yo Ehara. 2021. Lurat: a lightweight unsupervised automatic readability assessment toolkit for second language learners. In *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 806–814.

Yo Ehara. 2022. Selecting reading texts suitable for incidental vocabulary learning by considering the estimated distribution of acquired vocabulary. In *Proceedings of the 15th International Conference on Educational Data Mining, EDM 2022, Durham, UK, July 24-27, 2022*. International Educational Data Mining Society.

Yo Ehara. 2023. Innovative software to efficiently learn english through extensive reading and personalized vocabulary acquisition. In *Artificial Intelligence in Education. Posters and Late Breaking Results, Workshops and Tutorials, Industry and Innovation Tracks, Practitioners, Doctoral Consortium and Blue Sky - 24th International Conference, AIED 2023, Tokyo, Japan, July 3-7, 2023, Proceedings*, volume 1831 of *Communications in Computer and Information Science*, pages 187–192. Springer.

Yo Ehara, Issei Sato, Hidekazu Oiwa, and Hiroshi Nakagawa. 2012. Mining words in the minds of second language learners: Learner-specific word difficulty. In *Proceedings of COLING 2012*, pages 799–814, Mumbai, India. The COLING 2012 Organizing Committee.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Riku Hoshino and Masanori Morise. 2021. Neural reading and pronunciation prediction for japanese text-to-speech synthesis. *IEICE Transactions on Information and Systems*, E104-D(2):285–293.

Daisuke Kawahara and Sadao Kurohashi. 2013. Morphological analysis for unsegmented languages using recurrent neural network language model. *EMNLP 2013*, pages 1292–1302.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to japanese morphological analysis. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 230–237.

NAIST Computational Linguistics Lab. 2007. Ipadic: Mecab standard dictionary. https://github.com/taku910/mecab/tree/master/mecab-ipadic. Japanese dictionary for morphological analysis.

Ministry of Education, Culture, Sports, Science and Technology (MEXT). 2017. Curriculum guidelines "zest for life" appendix: Kanji allocation by grade. https://www.mext.go.jp/a_menu/shotou/new-cs/youryou/syo/koku/001.htm. Government of Japan White Paper.

Andrew Cameron Morris, Viktoria Maier, and Phil Green. 2004. From wer and ril to mer and wil: improved evaluation measures for connected speech recognition. *Eighth International Conference on Spoken Language Processing*.

Masaaki Nagata. 1999. A new japanese word segmentation method and its application to automatic furigana generation. *Natural Language Engineering*, 5(4):389–409.

R William Soukoreff and I Scott MacKenzie. 2003. Metrics for text entry research: An evaluation of msd and kspc, and a new unified error metric. *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 113–120.

Yu Tanaka, Yugo Murawaki, Daisuke Kawahara, and Sadao Kurohashi. 2021. Construction of japanese wikipedia input error dataset v2. In *Proceedings of the 27th Annual Meeting of the Association for Natural Language Processing*, pages 1001–1006. The Association for Natural Language Processing.

Google Brain Team. 2018. Tensorflow.js: Machine learning for javascript developers. https://www.tensorflow.org/js. JavaScript library for training and deploying ML models in the browser.

Kyoto University. 2011. Kyoto university web document leads corpus. https://nlp.ist.i.kyoto-u.ac.jp/index.php?KWDLC. Annotated Japanese corpus with morphological and dependency information.

Liang Wang and Philip C Woodland. 2003. Word level confidence annotation using combinations of features. In *Eighth European Conference on Speech Communication and Technology*.

Nan Xu, Fei Wang, Ben Zhou, Bangzheng Li, Chaowei Xiao, and Muhao Chen. 2024. Cognitive overload: Jailbreaking large language models with overloaded logical thinking. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3526–3548, Mexico City, Mexico. Association for Computational Linguistics.

Yusuke Yasuda, Xin Wang, Shinji Takaki, and Junichi Yamagishi. 2018. Neural machine translation with pronunciation prediction for japanese text-to-speech synthesis. In *Proceedings of Interspeech 2018*, pages 3267–3271.