

RUBY-ASR: EVIDENCE-PRESERVING SUPERVISION FOR JOINT ORTHOGRAPHIC AND LEXICAL-READING RECOGNITION

Hao Shi, Yun Liu, Xuehao Yang,
Jun Liu, Chuanbo Hua, Xuanjun Chen, Lianbo Liu, Shiao Zhu, Zixiong Su

Independent Researcher

hshi@ieee.org, liuyun5620472@gmail.com, sunskyxh@gmail.com
liujun@alu.uestc.edu.cn, cbhua@kaist.ac.kr, zxsu@g.ecc.u-tokyo.ac.jp

ABSTRACT

Conventional Japanese automatic speech recognition (ASR) is supervised by an orthographic transcript, although the same written form can correspond to different lexical readings realized in speech. Such utterances receive an identical target, so their reading distinction is absent from the supervision interface and cannot be recovered reliably by post-hoc text-only grapheme-to-phoneme conversion. We present **Ruby-ASR**, which refines the conventional target into a span-bound orthographic-lexical-reading sequence, e.g., 今日[きょう]. Unlike separate full-sentence orthographic and phonological outputs, the ruby representation locally binds each written span to its realized reading and permits deterministic recovery of both views. We instantiate the target under subtitle-style and verbatim-style transcription conventions using a Qwen3-ASR backbone; a mora-level CTC objective provides auxiliary monotonic reading supervision. The experimental results across five Japanese benchmarks show that refining the recognition target can improve lexical-reading recovery without sacrificing readable orthographic transcription. We release the checkpoints and inference code.

Index Terms— Japanese ASR, lexical-reading recognition, ruby annotation, orthography-phonology ambiguity

1. INTRODUCTION

Automatic speech recognition is conventionally formulated as mapping speech to an orthographic transcription. For Japanese, the output is a readable mixture of kanji and kana and is commonly evaluated using character error rate (CER) [1, 2]. However, it does not always identify how a lexical item was realized in speech. Japanese orthography and lexical reading are not in one-to-one correspondence: different written forms can share a reading, while one written form can admit multiple readings [3, 4, 5, 6]. If two utterances have different readings but the same orthographic transcript, orthographic ASR assigns them an identical target and does not require this distinction at the output. We call this *target-level lexical-reading collapse*. A downstream grapheme-to-phoneme (G2P) model cannot resolve the problem because it observes only the recognized text and infers a likely reading from linguistic context [7, 8, 9]. Likewise, text-only kanji-reading benchmarks [6, 10] cannot test whether a reading follows the input speech. Conversely, kana- or phoneme-only ASR preserves pronunciation but loses the orthographic distinctions and readability provided by kanji.

Joint phoneme-grapheme ASR has addressed this tradeoff using shared encoders and separate utterance-level decoders [11, 12, 13, 14]. More closely, Omachi et al. serialize each grapheme or multi-

grapheme span with its aligned phonemic annotation in a single sequence [15]. Transcript-prompted Whisper predicts Japanese phonemic and prosodic annotations from audio conditioned on a ground-truth transcript [16], but is an annotation model rather than speech-only ASR. The remaining challenge is to scale speech-conditioned lexical-reading supervision while retaining a directly usable orthographic transcript.

We therefore propose **Ruby-ASR**, which replaces the conventional target with a structured ruby-form sequence such as “今日[きょう].” This refines the output equivalence classes induced by orthographic supervision: utterances with the same surface form but different realized readings receive different labels. Its local span-reading binding permits deterministic recovery of both conventional orthographic and lexical-reading transcripts. We train two variants with the same architecture and objective: *Ruby-ASR-sub* follows a concise subtitle-style convention, whereas *Ruby-ASR-ver* follows a more literal verbatim-style convention. Ruby generation is the primary inference path, while a shared mora-level CTC branch provides auxiliary monotonic supervision and can optionally produce a decoder-free reading hypothesis.

Our contributions are threefold. First, we formulate lexical-reading loss as a target-level collapse under orthographic supervision and introduce an oracle-orthography G2P diagnostic that isolates the empirical text-only reconstruction gap from orthographic ASR errors. Second, we scale span-bound ruby supervision to a large LLM-based Japanese ASR system using an evidence-constrained target-construction pipeline. Third, evaluation on five Japanese benchmarks shows improved direct lexical-reading recognition with competitive orthographic accuracy; the checkpoints and inference code are publicly released¹².

2. BEYOND ORTHOGRAPHIC SUPERVISION

2.1. Target-level lexical-reading collapse

Let $\mathbf{X}$ denote an utterance, $\mathbf{O}$ its orthographic representation, and $\mathbf{R}$ its lexical reading realized in speech. Conventional Japanese ASR learns

$$p(\mathbf{O} | \mathbf{X}). \quad (1)$$

For lexical reading, orthography is not always sufficient:

$$H(\mathbf{R} | \mathbf{O}) > 0, \quad I(\mathbf{R}; \mathbf{X} | \mathbf{O}) > 0. \quad (2)$$

For $\mathbf{O}_a = \mathbf{O}_b$ but $\mathbf{R}_a \neq \mathbf{R}_b$, orthographic supervision maps $\mathbf{X}_a$ and $\mathbf{X}_b$ to the same target, so the reading distinction is absent from

¹<https://github.com/hshi-speech/Ruby-ASR-1.7B>

²<https://huggingface.co/hshispeech/Ruby-ASR-1.7B>

Table 1. Comparison of orthography–reading supervision interfaces. “Orth.” and “Reading” denote supervised outputs; “Span-bound” denotes explicit correspondence between each O_j and R_j ; and “Speech for R ” denotes speech-conditioned reading prediction rather than post-ASR G2P. Separate sequence targets lack explicit span correspondence, whereas Ruby-ASR uses an interleaved span-bound target.

Interface	Orth.	Reading	Span-bound	Speech for R
Orthographic ASR	✓	–	–	N/A
ASR + G2P	✓	Inferred	–	No
Kana/phoneme ASR	–	✓	–	Yes
Separate sequence heads	✓	✓	–	Yes
Interleaved span target	✓	✓	✓	Yes

the learning interface. Equivalently, it induces

$$\mathbf{X}_a \sim_{\mathcal{O}} \mathbf{X}_b \iff \mathbf{O}_a = \mathbf{O}_b, \quad (3)$$

whereas Ruby supervision refines the relation to

$$\mathbf{X}_a \sim_{\text{ruby}} \mathbf{X}_b \iff (\mathbf{O}_a, \mathbf{R}_a) = (\mathbf{O}_b, \mathbf{R}_b), \quad (4)$$

thereby separating examples collapsed by orthographic supervision.

2.2. Supervision interfaces and prior joint targets

Table 1 compares the relevant supervision interfaces. Orthographic ASR preserves readability but omits the realized reading; ASR followed by G2P infers it only after speech has been reduced to text. Kana-only ASR directly recognizes reading but loses orthographic identity. Prior multi-target systems retain both views using separate utterance-level sequence heads [11, 13, 14], but do not explicitly bind their corresponding spans. Omachi et al. instead combine locally aligned grapheme, phoneme, and optional POS subsequences in one autoregressive target [15]. Ruby-ASR scales this closely related interface to LLM-based Japanese ASR as 今日[きょう]学校[がっこう]に行[い]く, with deterministic orthographic and reading projections. In contrast, transcript-prompted annotation requires ground-truth orthography at inference time [16].

2.3. Text-only reconstruction gap

For recognized orthography, post-hoc G2P predicts

$$\hat{\mathbf{R}}_{\text{ASR+G2P}} = \text{G2P}(\hat{\mathbf{O}}). \quad (5)$$

This conflates orthographic recognition errors with text-only reading reconstruction. We therefore also evaluate

$$\hat{\mathbf{R}}_{\text{oracle-G2P}} = \text{G2P}(\mathbf{O}^*), \quad (6)$$

where $\mathbf{O}^*$ is ground-truth orthography, and define

$$E_{\text{text}} = \text{CER}(\text{G2P}(\mathbf{O}^*), \mathbf{R}^*) \quad (7)$$

as the *empirical text-only reconstruction gap* of the selected G2P system, not a theoretical irreducible bound. To avoid circularity, the primary analysis uses only reading references that are manually verified or prepared independently of that G2P system.

3. RUBY-ASR

3.1. Speech–language backbone

Ruby-ASR is initialized from Qwen3-ASR [17]. Given speech $\mathbf{X}$, the acoustic encoder and modality projector produce

$$\mathbf{H}^a = \mathcal{E}_{\text{aud}}(\mathbf{X}), \quad \mathbf{Z}^a = \mathcal{P}(\mathbf{H}^a). \quad (8)$$

An autoregressive language-model decoder predicts

$$p(\mathbf{Y} | \mathbf{X}) = \prod_{n=1}^N p(y_n | \mathbf{Y}_{<n}, \mathbf{Z}^a). \quad (9)$$

Under conventional supervision, $\mathbf{Y} = \mathbf{O}$ and the realized reading is not part of the output target.

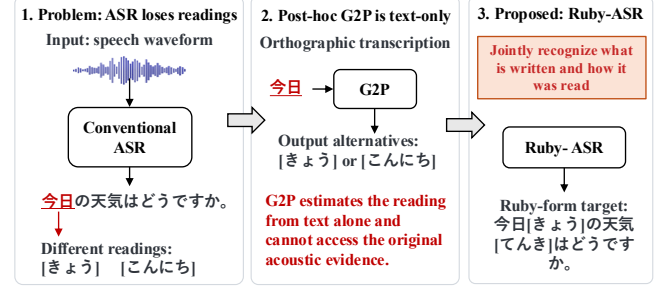


Fig. 1. Motivation and overview. Orthographic supervision assigns an identical target to acoustically distinct readings. Ruby-ASR instead binds each orthographic span to the reading realized in speech.

3.2. Span-bound Ruby target

Let O_j denote an annotated orthographic span, R_j its realized reading, and U_j unannotated material whose reading is already explicit, primarily kana. We serialize the Ruby target as

$$\mathbf{Y}^{\text{ruby}} = U_0 O_1 [R_1] U_1 \cdots O_J [R_J] U_J. \quad (10)$$

Readings are normalized to hiragana, and brackets are reserved structural tokens. Annotation is performed at lexical-span rather than character level, because compound readings are not necessarily compositional. For example, 今日[きょう] forms one pair, while okurigana remains outside the annotation, as in 行[い]く. A complete target may therefore be 明日[あした]は学校[がっこう]に行[い]く. This extends the interleaved transcription–annotation formulation of prior work [15] to Japanese lexical-span Ruby.

Both output views can be recovered deterministically:

$$\hat{\mathbf{O}} = \mathcal{P}_{\text{orth}}(\hat{\mathbf{Y}}^{\text{ruby}}) = U_0 O_1 U_1 \cdots O_J U_J, \quad (11)$$

$$\hat{\mathbf{R}} = \mathcal{P}_{\text{read}}(\hat{\mathbf{Y}}^{\text{ruby}}) = \rho(U_0) R_1 \rho(U_1) \cdots R_J \rho(U_J),$$

where $\rho(\cdot)$ normalizes unannotated material and removes non-spoken symbols. Thus, one generated sequence provides locally aligned orthographic and lexical-reading transcriptions. The Ruby sequence is trained autoregressively:

$$\mathcal{L}_{\text{ruby}} = - \sum_{n=1}^N \log p(y_n^{\text{ruby}} | \mathbf{Y}_{<n}^{\text{ruby}}, \mathbf{Z}^a). \quad (12)$$

This makes the realized reading an explicit speech-conditioned target. However, because R_j follows O_j , the decoder may also exploit preceding orthographic context; Ruby supervision preserves and binds reading information, but does not by itself prove causal reliance on speech.

3.3. Auxiliary mora-level regularization

To provide complementary monotonic reading supervision, the shared encoder is followed by additional Transformer layers [18, 19, 20, 21] and a mora-level CTC head:

$$\mathbf{H}^m = \mathcal{T}_{\text{mora}}(\mathbf{H}^a), \quad \mathcal{L}_{\text{CTC}} = -\log p_{\text{CTC}}(\mathbf{M} | \mathbf{H}^m), \quad (13)$$

where $\mathbf{M}$ is a mora-level reading sequence. Joint training uses

$$\mathcal{L} = (1 - \lambda) \mathcal{L}_{\text{ruby}} + \lambda \mathcal{L}_{\text{CTC}}. \quad (14)$$

Ruby generation remains the main inference path. The CTC branch can also produce an optional decoder-free kana hypothesis, but is treated primarily as auxiliary regularization in this work.

3.4. Automatic target construction

Most source corpora contain orthographic transcripts but no speech-grounded lexical readings. Pronunciation-annotated text corpora [5] and furigana-annotated speech resources [22, 23] provide valuable coverage, but do not supply matched annotations for the full large-scale ASR training pool. We therefore use evidence-constrained

closed-set annotation. Each transcript is normalized and morphologically segmented to identify kanji-bearing lexical units and separate surface okurigana. Multiple open-source ASR systems filter unreliable audio–text pairs, and an ensemble of three CTC models aligns reading evidence to the retained spans. For each O_j , OpenJTalk supplies a candidate set $\mathcal{C}(O_j)$; an open-source language model may resolve contextual ambiguity only by selecting $R_j \in \mathcal{C}(O_j)$, rather than freely generating a label. Nonempty, canonical-kana candidates consistent with the aligned utterance reading are serialized as $O_j[R_j]$; unresolved or structurally invalid samples are excluded from Ruby supervision.

4. EXPERIMENTAL SETUP

4.1. Training data and transcription conventions

We fine-tune Qwen3-ASR using Eq. (14) on a source pool of approximately 93.1 million audio–text segments (171.5 thousand hours) of read, spontaneous, expressive, and synthetic Japanese speech. It combines internally curated data with ReazonSpeech [24], Common Voice 26 [25], and selected JSUT subsets [26]. No evaluation audio from B5K, CSJ, Book, CV8, or TEDx is used for training. Synthetic speech increases coverage of medium- and high-difficulty lexical readings. We construct separate training sets for two transcription conventions using the same architecture and objective. *Ruby-ASR-sub* follows a concise subtitle-style convention similar to Whisper and ReazonSpeech. *Ruby-ASR-ver* follows Qwen3-ASR’s more literal convention and uses a substantially larger verbatim-style subset. We fine-tune all parameters with $\lambda = 0.3$. The CTC branch predicts 278 mora-level classes through a two-layer, eight-head Transformer adapter. Training uses length-grouped dynamic batches with a 600-s padded-audio budget and bfloat16 AdamW with a peak learning rate of 2×10^{-5} , cosine decay, and 1% linear warm-up. Starting from an earlier fine-tuned checkpoint with a reinitialized optimizer, we train for one epoch (185,186 steps), evaluate every 1,000 steps, and select the checkpoint with the lowest development loss.

4.2. Benchmarks and baselines

We evaluate on JSUT-BASIC5000 (B5K), the Corpus of Spontaneous Japanese (CSJ) [27], JSUT-Book (Book), Common Voice 8 (CV8), and TEDx [28]. Baselines are Whisper-Large-v3 (Whisp.) [29], Kotoba-v2 (Kotoba) [30], NeMo- and k2-based ReazonSpeech systems (Rz-NeMo and Rz-k2), Qwen3-ASR (Qwen) [17], and Kana-Whisper (K-Whisp.) [6].

4.3. Metrics

References and hypotheses are consistently normalized for numerals, long vowels, sokuon, punctuation, ruby marks, and kana variants. **Raw CER** evaluates script-preserving orthographic transcription, whereas **SA-CER** compares kana regions by reading and kanji regions by orthography, reducing penalties from kana–kanji variation. **Kana CER** evaluates lexical readings using Ruby-ASR’s direct reading projection or a shared G2P for orthographic baselines. Oracle G2P applies the same system to ground-truth orthography. To avoid circularity, primary oracle results are limited to B5K and CSJ, whose reading references were independently verified. For correctly recognized orthographic spans, we define

$$\mathcal{J}_{\text{corr}} = j : \hat{O}_j = O_j^*, \text{ C-KanaCER} = \frac{\sum_{j \in \mathcal{J}_{\text{corr}}} d_{\text{edit}}(\hat{R}_j, R_j^*)}{\sum_{j \in \mathcal{J}_{\text{corr}}} |R_j^*|}. \quad (15)$$

We report it for all eligible spans and the multi-reading subset $|\mathcal{C}(O_j^*)| > 1$, together with span-level exact-reading accuracy

Table 2. Raw CER and script-aware CER (SA-CER, %) on five benchmarks. W.Avg. is character-count weighted. Lower is better.

Model	B5K	CSJ	Book	CV8	TEDx	W.Avg.
Raw CER						
Whisp.	7.20	19.77	20.28	8.27	9.52	11.36
Kotoba	8.98	18.95	22.63	9.39	10.75	12.33
Rz-NeMo	7.38	19.24	20.75	8.78	10.66	11.79
Rz-k2	6.52	16.82	20.73	7.64	9.09	10.35
Qwen	8.71	14.55	23.70	9.18	9.08	10.78
Ruby-ASR-ver	8.23	8.23	23.50	8.12	8.90	9.10
Ruby-ASR-sub	7.84	6.33	23.21	7.25	9.06	8.51
SA-CER						
Whisp.	5.43	18.89	7.74	5.36	8.38	9.34
Kotoba	7.16	17.97	9.66	6.59	9.62	10.27
Rz-NeMo	6.00	18.36	11.03	6.66	9.67	10.17
Rz-k2	5.02	15.77	9.85	5.20	7.96	8.51
Qwen	6.86	13.37	9.54	6.28	7.88	8.59
Ruby-ASR-ver	5.95	6.55	8.67	4.78	7.69	6.59
Ruby-ASR-sub	6.03	6.34	8.80	4.65	7.95	6.64

Table 3. Kana CER (%) for lexical-reading transcription. Ruby-ASR and Kana-Whisper are scored from direct reading outputs; the remaining orthographic systems use a shared G2P. * marks manually verified reading references.

Model	B5K*	CSJ*	Book	CV8	TEDx	W.Avg.
Whisp.	2.22	14.63	4.76	2.83	6.88	6.57
Kotoba	3.34	13.37	6.65	3.60	7.74	7.08
Rz-NeMo	1.89	12.65	5.85	3.18	6.88	6.17
Rz-k2	1.78	11.61	5.29	2.71	6.31	5.64
Qwen	3.07	9.90	6.85	3.50	6.14	5.74
K-Whisp.	0.87	4.39	5.39	5.27	7.31	4.66
Ruby-ASR-CTC	1.95	11.19	5.72	9.06	8.26	7.31
Ruby-ASR-ver	1.08	5.41	5.70	2.57	5.09	3.75
Ruby-ASR-sub	1.32	5.37	5.84	2.63	5.66	4.01

($\hat{R}_j = R_j^*$). Rare and unseen subsets are defined by the training frequency $f(O_j^*, R_j^*)$ of each exact orthography–reading pair: $1 \leq f < 10$ and $f = 0$, respectively; evaluation data are excluded from these counts.

5. RESULTS

5.1. Orthographic and script-aware recognition

Table 2 shows no tradeoff in readable transcription: Ruby-ASR-sub achieves the lowest weighted Raw CER (8.51%), and Ruby-ASR-ver the lowest weighted SA-CER (6.59%), compared with 10.78% and 8.59% for Qwen3-ASR. On JSUT-Book, Ruby-ASR-sub decreases from 23.21% Raw CER to 8.80% SA-CER, indicating that much of the raw error reflects script realization rather than recognized-reading mismatch.

5.2. Lexical-reading recognition

Ruby-ASR-ver and Ruby-ASR-sub obtain weighted Kana CERs of 3.75% and 4.01%, compared with 5.74% for Qwen3-ASR, 5.64% for Rz-k2, and 4.66% for Kana-Whisper. Ruby-ASR-ver outperforms the standalone mora-CTC path on every benchmark, reducing weighted error from 7.31% to 3.75%, consistent with treating CTC as auxiliary monotonic supervision rather than a replacement for contextualized Ruby generation. Although direct Ruby prediction performs best in aggregate, Kana-Whisper remains best on B5K and CSJ, and Whisper+G2P on JSUT-Book. These results do not establish whether the AR decoder uses speech evidence beyond preceding orthographic context; answering that question requires a controlled intervention outside this paper’s scope.

Table 4. Oracle-orthography G2P diagnostic on independently verified B5K and CSJ references. Values are Kana CER (%). Oracle and Qwen hypotheses use the same G2P and normalization; Ruby-ASR uses its direct reading projection.

Condition	Predictor input	B5K	CSJ
Oracle orth. + G2P	Ground-truth orth.	1.69	3.47
Qwen + G2P	Recognized orth.	3.07	9.90
Ruby-ASR-ver	Speech-conditioned	1.08	5.41
Ruby-ASR-sub	Speech-conditioned	1.32	5.37

Table 5. Results on G2P-hard utterances, defined by nonzero reading edit distance for oracle-orthography G2P. B5K contains 1,124/5,000 (22.5%) and CSJ 2,579/8,460 (30.5%) such utterances. Exact is utterance-level exact-reading accuracy (%).

Model	B5K		CSJ	
	Kana CER	Exact	Kana CER	Exact
Oracle orth. + G2P	6.07	0.0	7.65	0.0
Qwen + G2P	6.13	17.6	12.84	5.4
Rz-k2 + G2P	4.79	22.8	14.44	3.6
Ruby-ASR-ver	1.59	57.9	8.02	16.8
Ruby-ASR-sub	2.24	50.3	8.02	17.1

5.3. Oracle-orthography G2P diagnostic

Applying the same G2P to ground-truth orthography separates orthographic recognition errors from text-only reading errors. Oracle G2P in Table 4 retains 1.69% CER on B5K and 3.47% on CSJ, confirming a nonzero text-only reconstruction gap; relative to Qwen+G2P, it reduces error by 1.38 and 6.43 points, respectively, showing the additional effect of orthographic ASR errors. Ruby-ASR-ver and Ruby-ASR-sub outperform oracle G2P on B5K by 0.61 and 0.37 points but trail it on CSJ by 1.94 and 1.90 points, making the benefit domain dependent. This comparison is diagnostic rather than causal because the systems are unmatched and the Ruby decoder also observes preceding orthographic context.

5.4. G2P-hard reading analysis

Table 5 localizes gains to readings that the selected G2P cannot reconstruct from correct orthography. On B5K, Ruby-ASR-ver reduces Kana CER from 6.07% for oracle G2P and 4.79% for Rz-k2+G2P to 1.59%, with 57.9% exact-reading accuracy. On CSJ, oracle G2P has slightly lower CER than Ruby-ASR (7.65% vs. 8.02%), but the Ruby models more often recover the complete utterance reading (16.8–17.1% vs. at most 5.4% for ASR–G2P). Direct prediction therefore benefits B5K most clearly, whereas spontaneous CSJ retains a speech-recognition bottleneck. Because this subset depends on one G2P implementation, it is a targeted diagnostic rather than an intrinsic partition of reading difficulty.

5.5. Conditional and ambiguity-sensitive reading accuracy

Because overall Kana CER is dominated by text-predictable readings, we use Eq. (15) to evaluate correctly recognized orthographic spans. Table 6 shows that both Ruby variants exactly recover about 97% of readings when the orthographic span is correct. Multi-reading spans, roughly half of the eligible spans, retain similar accuracy, suggesting that dictionary-level ambiguity is not the dominant residual error. Degradation on rare and unseen pairs instead identifies lexical coverage as the main remaining challenge, especially on spontaneous CSJ. Because its rare and unseen subsets contain only 245 and 116 spans, these results are diagnostic. Conditioning on correct orthography prevents surface-recognition errors from being attributed to reading selection.

Table 6. Conditional reading results on correctly recognized orthographic spans. Each entry is C-KanaCER / exact-reading accuracy (%). Ranges cover Ruby-ASR-ver and Ruby-ASR-sub. The all-span counts are approximately 22.5k for B5K and 19.3k for CSJ; multi-reading spans comprise about 49%.

Subset	B5K	CSJ
All correct orth.	1.71–1.94 / 97.4	1.47–1.61 / 97.2
Multi-reading	2.8–3.7 / 96.8	2.3–2.8 / ~97
Rare ($1 \leq f < 10$)	5.3–5.5 / 86	30–32 / 23
Unseen ($f = 0$)	13.0–13.2 / 70	30–31 / 24

Table 7. Boundary overflow on the filtered ReazonSpeech test subset ($n = 3,431$). Start and End are normalized insertion rates (%) before and after the annotated subtitle span; Total is their sum. Utt. and ≥ 3 count utterances with any and at least three overflow characters, respectively; Max is the maximum overflow length. Lower is better.

Model	Start	End	Total	Utt.	≥ 3	Max
Whisp.	0.40	0.46	0.86	239	111	19
Kotoba	0.17	0.21	0.38	145	45	11
Rz-NeMo	0.73	0.22	0.95	296	123	19
Rz-k2	0.78	0.20	0.99	338	127	13
Ruby-ASR-sub	0.36	0.21	0.57	160	61	21

5.6. Subtitle-boundary diagnostic

Ruby-ASR-sub targets concise subtitle-style transcriptions restricted to the current segment. After numeral normalization and punctuation removal, we align each hypothesis and reference at the character level. Insertions before the first or after the last reference character define *start* and *end overflow*, respectively, and usually reflect neighboring speech captured near a segment boundary. Thus, boundary overflow measures segment-level output control rather than hallucination or recognition accuracy. Ruby-ASR-sub achieves the second-lowest total overflow (0.57%) and affects fewer utterances than Whisper, Rz-NeMo, and Rz-k2. Kotoba-v2 performs best, consistent with its ReazonSpeech fine-tuning and closer match to the reference conventions. Despite one 21-character outlier, Ruby-ASR-sub produces substantially fewer utterances with at least three overflow characters than the other non-Kotoba systems. Both ReazonSpeech baselines show more start overflow, suggesting greater sensitivity to preceding-segment speech. Overall, subtitle-style training improves boundary control. We treat this only as a boundary diagnostic because the references are insufficiently reliable for content-level scoring.

6. CONCLUSION

We presented Ruby-ASR, an evidence-preserving refinement of Japanese ASR supervision. Conventional orthographic targets can collapse different realized readings into one label, leaving post-hoc G2P to infer the reading from text alone; span-bound ruby targets instead retain orthographic identity and lexical reading in one structured sequence. Across five benchmarks, Ruby-ASR improves direct lexical-reading recognition while maintaining strong orthographic and script-aware transcription; a separate diagnostic shows that Ruby-ASR-sub generally restricts output to the annotated subtitle span. Oracle-orthography, G2P-hard, and conditional-reading analyses show that direct prediction is particularly effective on B5K cases where the selected G2P fails, while rare and unseen orthography–reading pairs remain a major limitation. The released models provide structured Ruby generation and optional decoder-free mora output for Japanese speech and reading-aware research.

7. REFERENCES

- [1] Shigeki Karita, Yotaro Kubo, Michiel Adriaan Unico Bacchiani, and Llion Jones, “A comparative study on neural architectures and training methods for japanese speech recognition,” in *Proc. Interspeech*, 2021, pp. 2092–2096.
- [2] Shigeki Karita, Richard Sproat, and Haruko Ishikawa, “Lenient evaluation of japanese speech recognition: Modeling naturally occurring spelling inconsistency,” in *Proc. CAWL*, 2023, pp. 61–70.
- [3] Yui Sudo, Kazuya Hata, and Kazuhiro Nakadai, “Retraining-free customized asr for enharmonic words based on a named-entity-aware model and phoneme similarity estimation,” in *Proc. Interspeech*, 2023, pp. 491–495.
- [4] Wen Zhang, “Pronunciation ambiguities in japanese kanji,” in *Proc. CAWL*, 2023, pp. 50–60.
- [5] Fumikazu Sato, Naoki Yoshinaga, and Masaru Kitsuregawa, “Building large-scale Japanese pronunciation-annotated corpora for reading heteronymous logograms,” in *Proc. 13th Lang. Resources Eval. Conf. (LREC)*, June 2022, pp. 7113–7121, European Language Resources Association.
- [6] L. Liu, S. Zhu, K. Washizaki, R. Yoneyama, H. Jeon, M. Zhao, Y. Fujita, H. Shi, N. Yoshida, Y. Gao, R. Koshkin, Y. Hono, and Y. Sudo, “Sarashina2.2-TTS: Tackling kanji polyphony in japanese speech generation via data scaling and targeted data synthesis,” *arXiv preprint arXiv:2606.25369*, 2026.
- [7] Yuhi Matogawa, Yusuke Sakai, Taro Watanabe, and Chihiro Taguchi, “Japanese rule-based grapheme-to-phoneme conversion system and multilingual named entity dataset with international phonetic alphabet,” in *Proc. SIGMORPHON*, 2024, pp. 77–86.
- [8] Sam Passaglia, “Yomikata: Heteronym disambiguation for Japanese,” Software, 2023, Version 0.0.4, accessed Aug. 27, 2026.
- [9] Tomoki Koriyama, “Benchmarking large language models for grapheme-to-phoneme conversion: A Japanese case study,” *arXiv preprint arXiv:2606.22009*, 2026.
- [10] Ryota Mibayashi, Hiroya Takamura, and Hitomi Yanaka, “YOMI-Bench: A benchmark for evaluating Kanji reading and phonological understanding of LLMs for Japanese,” *arXiv preprint arXiv:2607.00664*, 2026.
- [11] Yotaro Kubo and Michiel Bacchiani, “Joint phoneme-grapheme model for end-to-end speech recognition,” in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, 2020, pp. 6119–6123.
- [12] Hao Shi, Yuan Gao, Xugang Lu, and Tatsuya Kawahara, “Beyond Acoustic Prefixes: Persistent Grounding in Serialized Acoustic Memory for LLM-Based Multi-Talker Speech Recognition,” 2026.
- [13] Shreekantha A. Nadig, V. Ramasubramanian, and Sachit Rao, “Multi-target hybrid CTC-attentional decoder for joint phoneme-grapheme recognition,” in *Proc. IEEE Int. Conf. Signal Process. Commun. (SPCOM)*, July 2020, pp. 1–5.
- [14] Krishna D N, “A dual-decoder conformer for multilingual speech recognition,” *arXiv preprint arXiv:2109.03277*, 2021.
- [15] Motoi Omachi, Yuya Fujita, Shinji Watanabe, and Matthew Wiesner, “End-to-end ASR to jointly predict transcriptions and linguistic annotations,” in *Proc. 2021 Conf. North Amer. Chapter Assoc. Comput. Linguistics: Human Lang. Technol. (NAACL-HLT)*, June 2021, pp. 1861–1871.
- [16] Rui Hu, Xiaolong Lin, Jiawang Liu, Shixi Huang, and Zhenpeng Zhan, “Transcript-prompted Whisper with dictionary-enhanced decoding for Japanese speech annotation,” in *Proc. Interspeech*, 2025, pp. 2540–2544.
- [17] X. Shi, X. Wang, Z. Guo, Y. Wang, P. Zhang, X. Zhang, Z. Guo, H. Hao, Y. Xi, B. Yang, J. Xu, J. Zhou, and J. Lin, “Qwen3-ASR technical report,” *arXiv preprint arXiv:2601.21337*, 2026.
- [18] H. Shi, Y. Fujita, T. Mizumoto, L. Liu, A. Kojima, and Y. Sudo, “Serialized output prompting for large language model-based multi-talker speech recognition,” in *Proc. ASRU*, 2025, pp. 1–8.
- [19] J. Zhao, H. Shi, C. Cui, T. Wang, H. Liu, Z. Ni, L. Ye, and L. Wang, “Adapting Whisper for code-switching through encoding refining and language-aware decoding,” in *Proc. ICASSP*, 2025, pp. 1–5.
- [20] H. Shi, Y. Gao, Z. Ni, and T. Kawahara, “Serialized speech information guidance with overlapped encoding separation for multi-speaker automatic speech recognition,” in *Proc. SLT*, 2024, pp. 193–199.
- [21] H. Shi, Y. Fujita, R. Koshkin, M. Zhao, Y. Gao, L. Liu, and Y. Sudo, “Distilling LLM semantic priors into encoder-only multi-talker ASR with talker-count routing,” *arXiv preprint arXiv:2603.10587*, 2026.
- [22] Fumikazu Sato, Naoki Yoshinaga, Masashi Toyoda, and Masaru Kitsuregawa, “Reconstruction of the Aozora Bunko furigana-annotated speech corpus using prompts and multiple speech recognition candidates,” in *Proc. 31st Annu. Meeting Assoc. Natural Lang. Process. (NLP 2025)*, Mar. 2025, pp. 3092–3096, (in Japanese).
- [23] National Diet Library, “Aozora Bunko furigana-annotated speech corpus,” Online dataset, Mar. 2025, Version 2, accessed Aug. 27, 2026.
- [24] Y. Yin, D. Mori, and S. Fujimoto, “ReasonSpeech: A free and massive corpus for japanese ASR,” in *Proc. NLP*, 2023, pp. 1134–1139.
- [25] R. Ardila, M. Branson, K. Davis, M. Henretty, M. Kohler, J. Meyer, R. Morais, L. Saunders, F. M. Tyers, and G. Weber, “Common voice: A massively-multilingual speech corpus,” in *Proc. LREC*, 2020, pp. 4218–4222.
- [26] R. Sonobe, S. Takamichi, and H. Saruwatari, “JSUT corpus: Free large-scale japanese speech corpus for end-to-end speech synthesis,” *arXiv preprint arXiv:1711.00354*, 2017.
- [27] K. Mackawa, “Corpus of spontaneous japanese: Its design and evaluation,” in *Proc. SSPR*, 2003.
- [28] Shintaro Ando and Hiromasa Fujihara, “Construction of a large-scale japanese asr corpus on tv recordings,” in *Proc. ICASSP*, 2021, pp. 6948–6952.
- [29] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, “Robust speech recognition via large-scale weak supervision,” in *Proc. ICML*, 2023, pp. 28492–28518.
- [30] Sanchit Gandhi, Patrick von Platen, and Alexander M. Rush, “Distil-whisper: Robust knowledge distillation via large-scale pseudo labelling,” *arXiv preprint arXiv:2311.00430*, 2023.