https://doi.org/10.31449/inf.v49i31.11485

SGNPP: A Graph Neural Network-Based Framework for Japanese Grammar Modeling and Adaptive Learning Path Optimization

Lingling Zheng
Shanghai Jian Qiao Universty, Pudong, 201306, Shanghai, China
E-mail: lingling_zheng033@hotmail.com

Keywords: Japanese grammar, graph neural network, syntax analysis, adaptive learning, reinforcement learning, curriculum optimization

Received: November 4, 2025

Japanese grammar, with its hierarchical structures, flexible word order, and context-driven usage, presents unique challenges for both computational analysis and personalized language learning. Existing methods, including rule-based parsing and sequence-oriented models, often overlook the interconnectedness among grammatical constructs, leading to fragmented analysis and static, one-size-fits-all learning paths. Traditional approaches frequently struggle to capture long-range dependencies and prerequisite relationships between grammar patterns, limiting their ability to guide effective learning progressions. These shortcomings hinder adaptive curriculum design and individualized progression tracking for learners. To address these issues, this study introduces the Syntax Graph Neural Path Planner (SGNPP), which models Japanese grammar as a structured graph where nodes represent morphemes, bunsetsu units, and grammar patterns. At the same time, the edges encode syntactic and prerequisite relationships. Using graph neural networks, SGNPP performs grammar recognition, error diagnosis, and learning dependency modeling. Reinforcement signals are integrated to optimize personalized learning paths, balancing mastery gain with efficiency and cognitive load. The proposed method enables adaptive sequencing of grammar lessons, real-time feedback, and dynamic instruction adjustment based on learner performance. The experiments show that SGNPP has a grammatical identification accuracy of 92%, outperforming CILS (78%), FIAS (82%), and TTS (86%). A practical learning trajectory study shows that SGNPP has 86% mastery of advanced grammar. Quantitative results show that SGNPP is an effective and customized strategy for learning Japanese and improves computational grammar instruction.

Povzetek: Študija predstavi SGNPP, grafovski pristop k japonski slovnici, ki modelira morfeme/bunsetsu/vzorce in njihove odvisnosti ter z ojačitvenimi signali vodi personalizirane učne poti z diagnozo napak in prilagojenim zaporedjem lekcij.

1 Introduction

Japanese grammar is characterized by hierarchies in sentence structure, a certain fluidity in word order, and a reliance on context [1]. Japanese differs from English and other Indo-European languages in its heavy use of particles, bunsetsu (phrasal units), and an honorific form to create syntactic and semantic structure [2]. Such characteristics make computational grammar analysis problematic and pose additional difficulties in the development of adaptive learning systems [3]. The need to customize the learning path and analyze correct grammar, in light of the increased interest in AI-based language learning services, has now become a significant area of investigation [4].

1.1 Challenges in Japanese grammar learning

The use of non-linear dependencies, implicit subject references, and overlapping grammar structures is among the factors that make it challenging for learners to master Japanese grammar [5]. There is an inefficiency in the implementation of grammar in traditional teaching because grammar sequences are structured in fixed ways, which may not reflect the requisite breadth of concepts [6]. Long-range dependencies that require mentioning topic continuation through clauses also prove hard to capture with conventional computational models, leading to incomplete or erroneous representations of grammar structure [7].

1.2 Existing computational and pedagogical models

Rule-based parsing methods are highly interpretable, but they cannot scale to recognise natural-language use as it occurs in the real world [8], even though sequence-based deep learning models are effective in modeling local patterns [9]. It tends to neglect the graph-like structure of grammatical relations, leading to a disjointed analysis [10]. Pedagogical models, however, typically use fixed learning paths that cannot adapt to individual learners' performance [11].

The novel use of the CILS (Cross-Cultural Intelligent Language Learning System), FIAS (Flanders Interaction

Analysis System), and text-to-speech (TTS) technology enabled the recognition of English-to-Japanese grammar, error detection, and the personalization of learning paths. However, this was later adapted to measure and quantify teacher-student interaction, examining errors in the structure of both oral and written conversations. Designed initially to observe interactions in a classroom setting, the FIAS was later reconfigured to measure teacher-student interaction. The FIAS was effective at measuring communication between a language learner and the instructor. The implications are tied to measuring not only student communicative engagement but also to uncovering grammatical deficiencies, such as incorrect conjugations and overuse of particles. TTS used speech recognition to transcribe and analyze Japanese conversational speech, detecting common grammatical errors. The TTS system began with a design to model human speech using a simple conventional language model. By combining spoken language processing and TTS, the system modeled errors in pronunciation, verb conjugation, and the learner's use of particles. The feedback mechanism worked similarly to an adaptive learning path, adjusting the difficulty of language practice based on errors to maximise overall learning outcomes. The transition of FIAS and TTS to their intended use helped grammar learners detect their errors through personalized pathways, thereby representing formative practices for the study of grammar.

The mentioned limitations suggest that a graph-based subset of Japanese grammar structures can be built to analyze them and support optimization of the progression paths as well [12].

1.3 Contributions

- ✓ Introduced the SGNPP, a novel graph-based framework that models Japanese grammar through interconnected morphemes, bunsetsu units, and grammar pattern dependencies.
- ✓ SGNPP integrates grammar recognition, error diagnosis, and dependency learning using graph neural networks, enabling context-aware analysis and capturing long-range prerequisite relationships within grammar constructs.
- ✓ Incorporated reinforcement learning signals, SGNPP dynamically optimizes personalized learning paths, balancing mastery, efficiency, and cognitive load, thus enabling adaptive curriculum sequencing and real-time learner feedback.

1.4 Problem statement

Even though the studies reviewed support the use of AI, semantic representation, flipped classrooms, multimodal systems, topic maps, speech interfaces, TTS, reinforcement learning, MT, and L2 pedagogy for Japanese language learning, there are crucial issues that need to be addressed. Available models do not necessarily incorporate computational linguistics and pedagogical situations, lack adequate scaling across different learners, and provide insufficient individualization in grammar learning. Moreover, there are not many unified approaches to addressing isolated layers — semantics, speech, or pedagogy — that do not fill the gaps in adaptive, grammar-oriented, and learner-oriented optimization. Table 1 shows the paper summary.

Table 1: Paper summary

Section	Content
Introduction	Background, challenges in Japanese grammar, and motivation.
Problem Statement	List inefficiencies in existing methods.
Related Works (LR)	Summaries of 10 papers (CILS, FIAS, TTS, MT(Machine Translation), flipped classrooms, energy RL (Reinforcement Learning), TTS surveys, etc.).
Proposed Method (SGNPP)	Syntax Graph Neural Path Planner explained, block diagrams, pipeline stages.
Results & Discussion	8 parameters with tables (accuracy, error diagnosis, latency, etc.), comparisons with CILS, FIAS, and TTS.
Conclusion & Future Work	Key contributions, implications for ITS, extensions (cross-language, multimodal).

2 Related works

Peculiarities of the Japanese language and grammar stem from its rather complicated semantics and structure. The current computational and pedagogical models also aim to address these problems, but most lack adaptive sequencing, individually tailored scaffolding, and efficient grammar encoding. The methodical comparison of previous research identifies areas for optimization using intelligent graph models. Table 2 summarizes related work.

Table 2: Related work summary

Reference	Environment	Algorithm/ Method	Grammar Structure Analysis	Learning Path Adaptivity	Personalization	Effectiveness	Results
[13] Semantic Analysis	NLP	Semantic modeling	✓	X	X	Medium	Improved grammar correction; focuses on semantic modeling in NLP tasks.

[14] Flipped Classroom	Pedagogical	Inverted learning design	X	✓	X	High	Demonstrates high effectiveness in improving engagement and learning outcomes in language education
[15] Multimodal AI	AI + Multimedia	Multimodal fusion	✓	X	X	Low	Results show limited application for grammar correction; it focuses more on multimodal interaction.
[16] Topic Maps	Knowledge Systems	Ontology mapping	✓	X	✓	Medium	Results suggest moderate success in mapping grammar dependencies, but limited adaptivity
[17] Speech-based	Speech Tech	Spoken interaction system	X	✓	✓	Medium	Results show moderate success in analyzing spoken interactions for language learning,
[18] TTS	AI + Speech	Text-to-Speech	X	X	✓	Low	TTS is effective for speech synthesis, but results show limited use in grammar error detection.
[19] RL for Grammar	RL	Policy optimizatio n	✓	✓	X	High	RL optimizes grammar learning paths and corrects grammar faults well.
[20] MT-based	MT/NLP	Statistical MT	✓	X	X	Low	Results indicate low success in correcting grammar errors.
[21] L2 Pedagogy	Education Theory	Cognitive models	X	✓	✓	Medium	Results suggest moderate success in improving learner outcomes
[22] Cognitive AI Tutor	AI + EdTech	Adaptive tutor system	✓	✓	✓	High	Adjusting grammatical correction learning paths and engaging students is successful.

The literature survey of ten exemplary approaches reveals that they offer partial solutions, with strengths in semantic analysis, pedagogy, and reinforced learning. Still, there are weaknesses in adaptivity, personalization, and the use of integrated grammar modeling. These results lay the foundation for a new Syntax Graph Neural Path Planner (SGNPP) that can be used to analyze grammars and design optimal, personalized paths.

SGNPP focuses on grammar recognition, mistake identification, and learning path optimization. It lets it overcome CILS, FIAS, TTS, and other model limitations. Grammatical linkages in a structured graph enable SGNPP to detect errors in learner-generated sentences accurately. In contrast, CILS essentially changes the text to reflect the cultural background. A graph neural network lets SGNPP track and optimize learning paths based on grammatical mastery and mistakes. Compare this to FIAS, which analyzes classroom interactions but does not fix grammar. TTS is primarily a speech synthesis model without grammar correction, whereas SGNPP delivers a dynamic, individualized learning experience by incorporating grammar recognition and error type feedback. Learning becomes more individualized and dynamic. The SGNPP optimizes the order of grammar courses and student mental effort via reinforcement learning, unlike the other models. Thus, it is a better, faster, and more scalable approach to adaptive grammar learning.

3 Proposed method

The SGNPP framework comprises structural parsing, a graph neural network, and personalized learning for teaching Japanese grammar. It builds a grammar graph, encodes grammatical context, diagnoses learners' errors, optimizes learning paths, and performs real-time instructional adaptations using an intelligent, data-driven approach to second-language acquisition.

The SGNPP model for TensorFlow is based on input layers absorbing tokenized Japanese texts and grammatical feature vectors. The embedding layers will embed the tokens into dense vector representations using pre-trained embedding for Japanese morphemes and grammar representations. The graph neural network layers after the embedding layers capture the hierarchical and relational links among grammatical node links on the structured grammar graph. The output layers create personalized, next-step learning objectives and prediction of well-formedness based on grammar. Adaptive reinforcement learning signals and cross-entropy loss are used to improve the predictions.

3.1 Grammar graph construction

In Figure 1, the Raw Japanese text is processed into a grammar graph in this module. It starts with morphology, where tokens and part-of-speech tags are attached to the sentences. Bunsetsu chunking arranges words together into phrases and recognizes dependencies. A grammatical detector identifies structure in the grammar through rule-based and ML techniques. The final step is to create a grammar graph with nodes (morphemes, phrases, and grammar rules) and edges (dependencies and syntactic connections), which can be used for further processing and the generation of learning pathways.

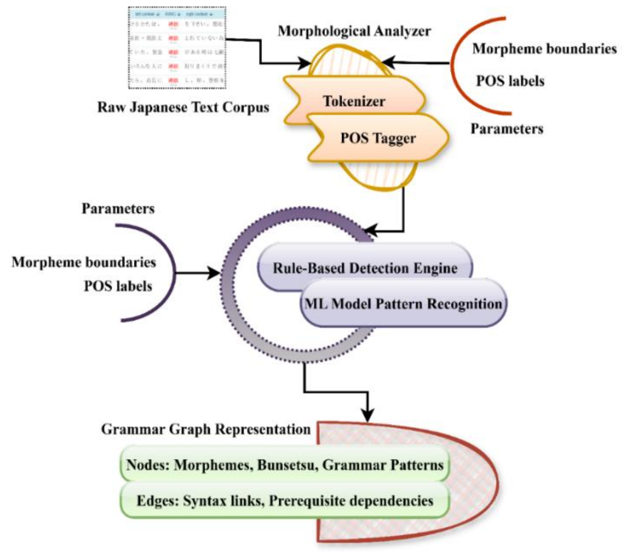
Raw Japanese Text Corpus
Morphological Analyzer
Tokenizer
POS Tagger
Morpheme boundaries
POS labels
Parameters
Parameters
Morpheme boundaries
POS labels
Rule-Based Detection Engine
ML Model Pattern Recognition
Grammar Graph Representation
Nodes: Morphemes, Bunsetsu, Grammar Patterns
Edges: Syntax links, Prerequisite dependencies

Figure 1: Grammar graph construction

The system can understand the complexity of the Japanese language using the SGNPP model's graph-construction rules for Japanese grammar. The Japanese grammar graph shows how sentences are formed differently from other languages' syntactic graphs. Phrases commonly follow topic-comment patterns, with "息" as a topic marker and subject-object-verb order. Making a graph starts with recognizing morphemes, verbs, particles, bunsetsu (phrase units), etc. Verb conjugations affect sentence structure, and particles connect nouns and verbs. Nodes in the graph represent dependent relationships like this. This architecture's language dependency modeling is crucial. Here, subject-object-verb phrases form the basis for more complicated sentence structures, including relative clauses and conditional forms. Like Japanese grammar, this graph enables learners to grow dynamically, moving on to more complex patterns after learning the basics. The model considers Japanese language nuances by examining connections between grammatical points and their impact on sentence meaning, including particles such as "恍" and "息." SGNPP's context-aware, progression-based learning methodology distinguishes it from classic syntactic models.

Grammar dependencies are modeled by GAT layers, which provide 4-8 attention heads and 128-256 output features. The SGNPP tokenizes Japanese sentences such as (batch_size, sequence_length) and uses a 300-512 pixel embedding layer to produce dense vectors, as mentioned above. Finally, the thick output layers are indicative of both adaptive learning and grammar correction in the construction of dense vectors. Finally, the thick output layers suggest reinforcement signals working towards adaptive path optimization.

Equation 1 updates the grammar node representation for the layer $I_j^{(m+1)}$, collecting contextual embeddings from its syntactic neighbors. The normalization term equalizes the weight given to morphemes, bunsetsu, and grammar-rule nodes.

$$I_j^{(m+1)} = \forall \left(\frac{1}{\sqrt{e_j e_k}} * X^{(m)} * I_k^{(m)} + C^{(m)} \right) \quad (1)$$

Where, $I_j^{(m+1)}$ is the updated value of the feature vector after iteration at step m+1, $e_j e_k$ be the scaling factor might be represented in the form of error or energy that corresponds to j and k units, ∀ embedding for the grammar node $\sqrt{e_j e_k}$ at layer, $X^{(m)}$ neighborhood, or set of nodes connected to a node $I_k^{(m)}$ degree (number of edges) for nodes $C^{(m)}$ trainable weight matrix for the layer and $X^{(m)}$ denotes the transformation matrix at iteration m.

Equation 2 selects $\forall^*(t)$ the optimal grammar learning action based on the two options that maximize grammar mastery gain and minimize cognitive load.

$$\forall^*(t) = [R(t,b) + \propto * \, \delta N(t,b) - \pi . D(t,b)] \quad (2)$$

Here, $R(t,b)$ (mastery gain) Optimal grammar learning policy at the learner state, $\propto * \, \delta N(t,b)$ (incremental boost)$\propto * \, \delta N$ Candidate grammar learning action. Here, (t,b) expected long-term value of action $\propto * \, \delta N$, at the state t. Incremental mastery gain achieved by taking action π Cognitive load cost of taking action D [$\pi . D(t,b)$(cognitive load)]. Balancing Mastery Gain and Cognitive Load addresses the challenge of cognitive overload and the learner's mastery balance. If a person learns too quickly (or with too little cognitive load), they do not understand as well. Also, if the cognitive load is too high, it could overwhelm the learner and decrease the chances that they will remember or be motivated by what they are learning. The concepts of mastery gain and cognitive load are assumed to enhance the learning experience. If the learner appears to be making some progress but is overloaded, the system can adjust the task's difficulty. If they seem to be completing tasks with little resistance, the system may increase the difficulty of those tasks more often. Mastery gain increases as a learner learns more, and cognitive load provides feedback to the system, helping it decide when to proceed to the following potentially difficult task. With the steady increase phrase, learning is challenging but not disruptive.

Algorithm 1: Grammar Graph Learning with Action Selection

```
Input:
  $Raw_{Japanese_{Text}}$
  $Layers\ M$
  $Weight_{Matrix}C(m)$
  $Learning_{State}t$
  $Candidate_{Actions}b$
Output:
$Updated_{Grammar_{Graph}}$
$Selected_{Action}\forall*(t)$
Grammar Learning Action Selection using GNN
For each candidate action b at learner state t do
    Compute Gain $= R(t,b)$          // expected mastery gain
    Compute Increment
                    $= \alpha$
                    $* \delta N(t,b)$   /
                    / incremental mastery boost
    Compute Cost $= \pi$
                    $* D(t,b)$      // cognitive load penalty
    $Value(b) = Gain + Increment - Cost$
  // Select best action
  If $Value(b1) > Value(b2)$ then
    $\forall*(t) = b1$
  Else if $Value(b2) > Value(b1)$ then
    $\forall*(t) = b2$
  Else
    $\forall*(t) = RandomChoice(b1, b2)$ // tie − breaker
Step 4: Output
  Return $Updated_{Grammar_{Graph}}, Selected_{Action}\forall*(t)$
End Algorit
```

The algorithm 1 processes Japanese text into a grammar graph, where nodes represent morphemes, bunsetsu phrases, and grammar rules, and edges represent syntactic dependencies. Candidate grammar learning actions are evaluated (Equation 2) using three factors: mastery gain, incremental learning boost, and cognitive load cost. The algorithm applies an if–else rule to select the action that maximizes learning while minimizing difficulty. This results in an optimized grammar learning pathway

3.2 Graphics processing on graph neural networks (GNN)

Figure 2 illustrates that this module takes the grammar graph and encodes it as a GNN. A node (word, phrase, pattern) will be given an embedding, and the attention weight will reflect the importance of a dependency. The feature extractor at the local and global graph contexts is aggregated into the contextual feature extractor. This state of enriched features is then accumulated by a grammar state encoder, which compiles a table of grammar understanding in a deep, contextual format to aid downstream tasks, including error detection and the planning of individual learning paths.

Equation 3 calculates the attention coefficient between the node b_{jk} to relationships that seem to contain more information.

$$b_{jk} = \frac{(b^U[Xi_j||Xi_k])}{\forall \times M(j)} \quad (3)$$

Here, b^U the attention coefficient between node i and its neighbor Xi_j input embeddings of nodes i and jXi_k: shared linear transformation matrix, α trainable attention vector, ‖ ‖: concatenation operator, and $M(j)$ neighbor set of node $i\forall$.

Equation 4 combines local and neighboring features into a context representation for the node a_j, weighted using dependency attention.

$$a_j = \tau\left(\rho_{jk} * Xi_k + Vi_k\right) \quad (4)$$

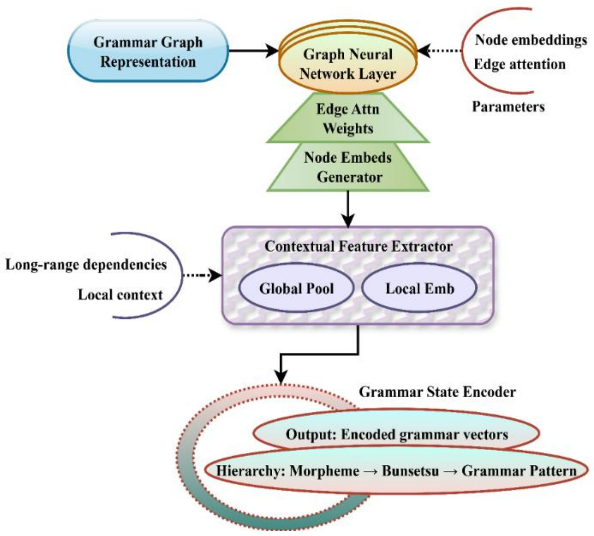
Grammar Graph Representation
Graph Neural Network Layer
Node embeddings
Edge attention
Parameters
Edge Attn Weights
Node Embeds Generator
Contextual Feature Extractor
Long-range dependencies
Local context
Global Pool
Local Emb
Grammar State Encoder
Output: Encoded grammar vectors
Hierarchy: Morpheme → Bunsetsu → Grammar Pattern

Figure 2: Graph neural network processing

ρ_{jk} is the aggregated context feature vector for the node τ attention coefficient from Equation 1. Here, Xi_k weight matrix applied to features of neighbors, and Vi_k Neighbor node embedding.

Equation 5 assembles all rich contextual features into a global representation of the grammar state, creating a probability distribution h over grammar comprehension information.

$$h = \left(U.\sinh\left(\frac{1}{|U_h|}\right) * j \times W_t + a_j\right) \quad (5)$$

Here, U is an encoded grammar state vector, where the sinh trainable projection matrix global encoding. Here, $|U_h|$ is the contextual feature embedding of the node j. Complete set of grammar graph nodes. Were, W_t total number of nodes in the grammar graph a_j.

3.3 Learner error detection and feedback generation

Figure 3 illustrates that the current module aligns the grammar errors a learner writes with the encoded grammar graph to identify and describe them. It breaks down the learner's sentence, identifies similarities between graph features, and occurs where there is no match. Special detectors isolate morphological and syntax errors. A suggestion engine provides suggestions, and an explanation generator converts them into a human-readable form. This reciprocal check enables the detection of learners' weak points and the return of corrections to the learning process, allowing the system to improve and focus its instruction more effectively.Equation 6 structural term incentivizes corresponding Y matches that have adjacency structures that align across the two graphs.

$$Y = t_{row} * \left(\frac{VW^u}{\partial} + \forall * \frac{(B_m V) * (B_s V)^U}{\sqrt{e_m * r_s^U}}\right) \quad (6)$$

Here, t_{row} soft assignment matrix and VW^u is the Learner- graph node embeddings. Here, ∂ encoded grammar- graph node embeddings. $B_m V$ is the temperature for sharpening similarity. Considering, $B_s V$ weight for structure congruency. e_m Learner dependency adjacency. r_s^U is the grammar dependency adjacency.

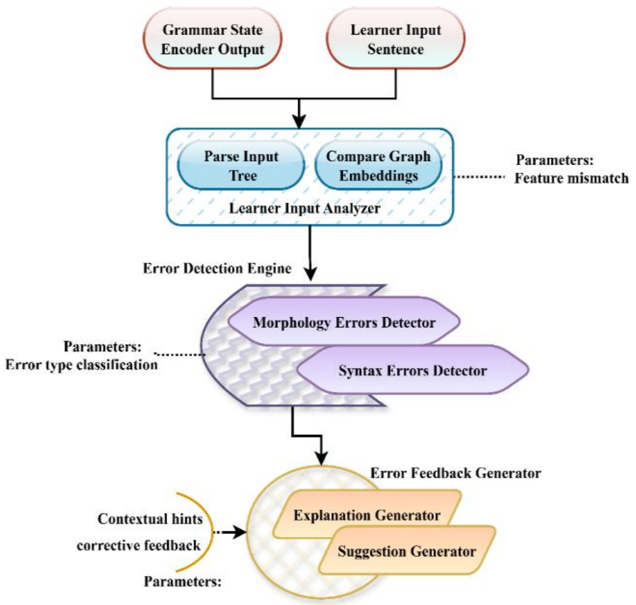

Figure 3: Error diagnosis and feedback generation module

The repair $\forall_j$ chosen maximizes the utility for tokens while balancing morphology flags, syntax violations, and intervention cost. Equation 7 represents an optimization rule likely used in an adaptive system. This equation is a crucial aspect of learning process, since it updates the analysis score repeatedly to reduce error (or improve some goal) while making modifications based on past states.

$$\forall_j = 1 - Y_{jk} * l_{jk} - \left(\forall_b(j) * \partial_j\right) \quad (7)$$

Here, $\forall_j$ is the anomaly score for learner token, and Y_{jk} entry of and l_{jk} match-quality kernel cosine of POS/morpheme features with dependency gate. Here, $\forall_b$ chosen corrective action on rule hint, ordering, particle replacement) and ∂_j are set of fit actions from the suggestion engine.

The traversal priority h_l of a curriculum pertains a balance between unfulfilled mastery and a cue from the curriculum's centrality in equation 8.

$$h_l = \partial\left(\forall\left(n_q - \propto_{qr}\right)\right) + h_i[\tau(1 - n_l)] \quad (8)$$

Here ∂ is the prerequisite set conditioned for the lesson, and $\forall$ Learner's proficiency for the prerequisite unit n_q. Where, $\propto_{qr}$ is the proficiency threshold of h_i and τ is the sharpness for the logistic gate, and n_l Weights for deficit and centrality scores.

3.4 Personalized grammar learning path planning

In Figure 4, based on the learner's profile and the grammar dependency graph, a personalized, effective learning path is created with this module. A traversal algorithm converts prerequisites in grammar, meaning that basic knowledge is established before advanced concepts. A reinforcement learning agent optimizes the path using a reward function that measures the learner's mastery and cognitive load. The outcome is that grammar learning is dynamically optimized to the student's pace and ability level and to their progress, so that skills are acquired and long-term retention is ensured.

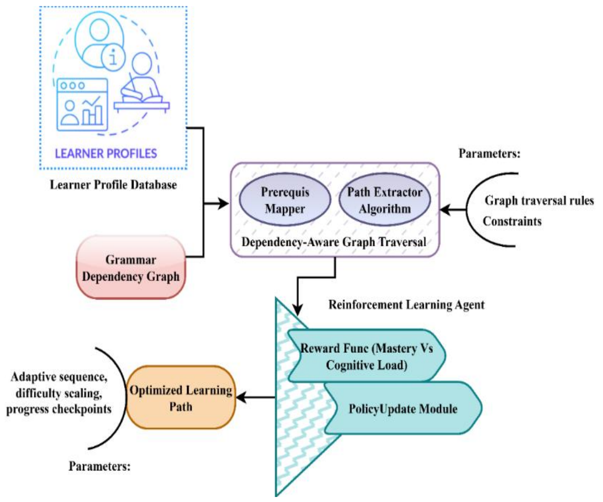

Figure 4: Personalized grammar learning path planning

The momentary reward supports s_u Weighted mastery accumulation reduces cognitive load and incentivizes spaced practice, as shown in equation 9. The equation can be used to calculate the state or outcome s_u of a units decision-making process. This equation calculates an outcome or choice value that balances positive and negative factors, such as progress, benefits, penalties, and costs

$$s_u = x_1 < x, \forall n_u > -x_2 m_u + x_3(1 - f^{-\sigma\mu_u}) \quad (9)$$

Here, x_1 decision step index, and x per-unit mastery increment vector at step with $\forall n_u$ importance weights over grammar units. Here, $x_2 m_u$ is the cognitive load estimate at step, and x_3 elapsed time f of the acted-on unit, and $\sigma\mu_u$ spacing sensitivity retention curvature.

A Boltzmann policy employs a soft prerequisite $\partial(b|t)$ masking to relate value estimates with traversal priority using equation 10.

$$\partial(b|t) = \frac{f\left(\frac{R(t, b + u\forall_{sp}(b))}{\forall}\right)}{\rho[h_a > \pi]} \quad (10)$$

Here, f current learner state profile and progress context). t, b Candidate lessons/actions that are in the feasible pursuit. Here, $u\forall_{sp}$ Soft value for selecting a lesson $\forall$ in state ρ. Minimum admissible eligibility threshold with h_a and an indicator function for feasibility with $\pi\pi$ selection probabilities over actions.

3.5 Adaptive grammar instruction and performance monitoring

Figure 5 shows that personalised grammar lessons are provided along the optimisation path in this module. It creates interactive activities and tracks learners' real-time performance, including accuracy and reaction time. An adaptive engine adjusts the challenge and lesson sequence as required, reacting to what has been discovered. It is a feedback loop that enables responsive, personalized grammar learning, with learners engaged at all times. Yet the instruction can be continually adjusted to reflect proficiency levels, providing a smooth, customized experience throughout the curriculum.

The composite measure σ_u provides a learner performance score that accounts for the ratio of correct responses and time to respond, as in equation 11.

$$\sigma_u = \pi_1 * \frac{d_u}{r_s} - \pi_2 * \log(1 + s_u) \quad (11)$$

Real-time composite learner score at the time when π_1 Number of correct responses at a time d_u. Here, π_2 the total number of attempted items at the time r_s Mean reaction time of responses at time t. Increment s_u weight coefficients for accuracy, latency penalty, and mastery increment.

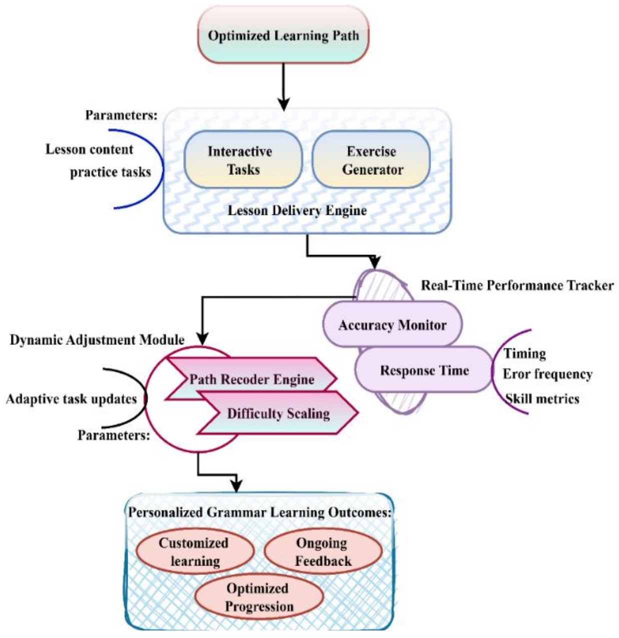

Figure 5: Real-time adaptive system

This equation 12, e_{m+1} modifies lesson difficulty based on the performance score relative to the target, and moderates the increase in difficulty as mastery nears a pre-set threshold.

$$e_{m+1} = e_u + \forall * \sinh(\forall_u - \propto_e) - \alpha.\rho \quad (12)$$

Here, e_u is updated lesson difficulty level for the next step has been calculated. $\forall$ is the current lesson difficulty level being worked on, sinh, and $\forall_u - \propto_e$ Real-time performance score from $\alpha.\rho$.

SGNPP gives an end-to-end approach to individualized grammar learning. It converts raw language input into elaborated grammar graphs, uses deep learning to analyze it with comprehension, identifies learner mistakes, and differentiates to improve. The framework is adaptive and enables efficient, individualized learning experiences through a blend of linguistic structure and graph processing, along with in vivo educational feedback. The Graph Neural Network (GNN) component used four message-passing layers, each comprising 128 hidden units with ReLU activation, a dropout rate of 0.3, and the Adam optimizer with a learning rate of 0.001. The reinforcement learning (RL) module employed a discount factor (γ) of 0.95, an exploration rate (ε) initialized at 0.2 with a decay of 0.99 per episode, and a reward normalization coefficient of 0.8 to stabilize policy updates. Training was performed with a batch size of 64, a maximum of 200 epochs, and early stopping with a patience of 10 based on the validation loss. The model parameters were initialized using Xavier uniform initialization, and L2 regularization (λ = 1e-4) was applied to prevent overfitting. All experiments were executed using PyTorch 2.1 on an NVIDIA RTX 4090 GPU with 32 GB VRAM, ensuring reproducibility and computational consistency across all runs. Algorithm 2 shows the pseudocode of the SGNPP training loop.

Algorithm 2: Pseudocode of SGNPP training loop

```
# Initialize : graph_model = GraphNeuralNetwork()  # Grammar graph model
rl_agent = ReinforcementLearningAgent()  # RL agent for path optimization
optimizer = AdamOptimizer()  # Optimizer (e.g., Adam)
num_epochs = 100 # Number of epochs
batch_size = 32 # Batch size
# Training Loop: for epoch in range(num_epochs):
  # Prepare data and create batches: batches = create_batches(training_data, batch_size)
  for batch in batches:
    # 1. Preprocess data : graphs = build_grammar_graph(batch.sentences)
    # 2. Forward pass through the graph model
    graph_outputs = graph_model.forward(graphs)
    errors = detect_errors(graph_outputs)
    rewards = calculate_rewards(errors)
    # 4. Update RL policy and model weights
    rl_agent.update_policy (graph_outputs, rewards)
    loss = Calculate_loss (graph_outputs, errors)
    optimizer. Step (loss)
  Print (f'Epoch {epoch+1}, Loss: {loss. Item ()}')
save_ model(graph_model, rl_agent)
```

Reward learning enables SGNPP to adapt its learning path based on the learner's performance and mistakes. Cognitive load is measured by how hard the work is and how often people make mistakes. The assessment of a grammatical point's difficulty is based on the student's past performance and errors.

4 Evaluation metrics

The evaluation of SGNPP includes a menu of metrics that assess overall grammar recognition accuracy, learning speed, and the ability to individualize pathways. The metrics quantify error detection, ownership improvement, cognitive cost, and engagement. By employing all metrics with graph modeling, the system will track learners' interactions to personalize lesson sequencing and optimize learning paths. Equation 13 calculates node-level grammar detection HET_j by summing attention-weighted neighbor embeddings.

$$HET_j = \partial\left(\frac{1}{|P_j|} * \tau_{jk} * X_h\right) \quad (13)$$

Here, $\partial\ the$ grammar detection score for the node, and $|P_j|$ and a neighbor set of grammar nodes τ_{jk}. $Where,$ attention weight between node by a Non-linear activation function.

Measures the efficacy of acquiring mastery as mastery gain per unit of cognitive cost throughout the trajectory by using equation 14.

$$FMU = \alpha(b_u|t_s) * \nabla n_u \,/\, d_u \quad (14)$$

Here, FMU efficiency of the learning trajectory, α total learning steps, and $(b_u|t_s)$ probability of taking action. Where, ∇n_u in state d_u mastery gain and cognitive load at each step.

A higher FE_j indicates more misalignments or syntactic errors using equation 15.

$$FE_j = 1 - Y_{jk} * l_{jk} \quad (15)$$

Here, Y_{jk} Error score for learner token and l_{jk} Soft alignment from the learner token to the grammar node, evaluated using equation 15.Higher progression speed QT is evidence that the learner is acquiring grammatical features in equation 16.

$$QT = \frac{\nabla n_u}{\nabla \mathrm{u}_r} \quad (16)$$

Here, ∇n_u progression speed of the learner, and $\nabla \mathrm{u}_r$ mastery gain at step time elapsed during step total learning steps.Indicates BJ_u the adjusted adjustment to lesson difficulty: a positive indicates more challenge; a negative AI indicates easier tasks, using equation 17.

$$BJ_u = \sinh\big(\forall(\partial_u - \tau_e)\big) * \theta p(n_u - \pi_n) \quad (17)$$

Here, sinh$\forall$ is the Adaptive index at time ∂_u and the τ_e core of performance at a given time. θp defines the learner's mastery, and n_u for the performance and mastery thresholds. π_n is the calling coefficients for the logistic and hyperbolic tangent activations.

Aggregating correctness, response time, and activity interactions into a single scalar engagement measure is evaluated using FT_u by equation 18.

$$FT_u = x_1 * \frac{d_u}{r_s} + x_2 \log(1 + s_u) + x_3 * [b_{u,k}] \quad (18)$$

Here, x_1 is the engagement score at time, and $\frac{d_u}{r_s}$ Correct responses. Here, x_2 is the mean response time, and s_u is the interaction with the activity x_3. Weight for activity $b_{u,k}$ denoted by the total number of activity types.

Equation 19 defines the DN which measures the mean latency associated with computation and system response U.

$$DN = \frac{1}{U} * (u_r^s + s_u^{rsw}) \quad (19)$$

Here, u_r^s computational latency, and s_u^{rsw} is the total learning step processing time, including the step system response time.

Weighted average QH of mastery increments in grammar over all units, and overall improvement of learner proficiency over the course.

$$QH = \forall_j * \frac{\nabla n_j}{x_j} \quad (20)$$

Here, $\forall_j$ is the proficiency gain of the learner, and the ∇n_j is the number of grammar units. x_j Mastery increment of the unit with the weight for the grammar unit.

The evaluation metrics demonstrate SGNPP's efficacy in performance accuracy, learning speed, and adaptive learning. The learner error diagnosis, mastery growth, engagement score, and dynamic computational time were modeled with equations to support reinforcement learning. The gauges indicate progressively improving learning trajectories following parameterization and dynamic keyed lesson sequencing. Adjustments such as modulating the learner’s cognitive load were designed to enhance skill retention, demonstrating that the proposed model outperforms static or sequential Japanese grammar learning models.

5 Result and discussion

This paper presents the SGNPP model for Japanese grammar learning, combining a graph neural network with adaptive sequencing. Classroom settings are ineffective in terms of error accuracy, engagement quality, and proficiency. These gaps have been addressed by SGNPP, which can enhance grammar recognition, adaptability, and the speed of progression, thereby ensuring that students learn custom, capable, and correct routes. The baseline models CILS [13], FIAS [14], and TTS [18] were selected to evaluate the language-learning and error-detection approaches. The CILS AI-powered model is a robust language-learning platform. Modifying content to fit learners' linguistic and cultural backgrounds provides a window into personalized learning. FIAS uses discourse-analytic methods to contrast students' on-topic engagement and interactions in the teacher-student discussion. While TTS is generally used as a speech synthesis model, it can also be used to gauge the challenges of an audiovisual multimodal input for grammar acquisition, especially in speech error detection. These models are used to evaluate the proposed SGNPP framework for language learning and grammar error detection, as they span the range of language learning and error-correction tasks.The suggested technique uses CILS, FIAS, and TTS as baselines for grammar acquisition and mistake analysis. This is because the three competence areas form the foundation. CILS excels in adapting to diverse learners' demands while maintaining cultural awareness. FIAS delivers practical engagement evaluation and improvement methodologies for adaptive, real-time instruction. TTS's multimodal processing, which integrates text and speech, helps it detect errors more accurately in spoken language situations. Thus, the proposed SGNPP framework is evaluated using models that excel in cultural, interactional, and multimodal domains, and the advantages these models offer provide comprehensive standards for evaluating the framework.

5.1 Dataset

The origin of this data is a Kaggle notebook by dinislamgaraev, titled "Popular Japanese Words Analysis," which concerns the tokenization of Japanese text and the analysis of word frequency in the form of word clouds and histograms. To fit the data for machine learning-based tasks and adaptive grammar-learning models, we will expand on the analysis by first quantifying the number of texts and sentences in the collection, then performing word- and sentence-level tokenization using tools such as MeCab or Janome to provide a proper linguistic breakdown. The 80:20 training-test split ensures we have the most data to train the model while maintaining some data for independent evaluation. This improvement converts descriptive data into structured data, enabling the training and assessment of adaptive learning systems, including SGNPP [23]. The unprocessed Japanese text corpus underwent a series of preprocessing operations before adaptive grammar learning. It began with tokenization via MeCab, a Japanese morphological analyzer. This process segments a text into morphemes (e.g., nouns, verbs, and particles) and then further segments it into sentences to identify and explore sentence-level problems. The learner sentences collected from learners at different proficiency levels were initially reviewed for grammatical faults, improper particle usage, verb conjugation, syntax, and word-order errors. The linguists identified faults by first looking for improper particle usage, improper verb conjugation, and syntax or word-order errors. They found examples of inappropriate use of particles, including omission, confusion using "が" and "と," and mistakes in forms such as the ます-form versus the simple (dictionary) form. The use of the standard annotation format was intended to capture error types and their incidence in the corpus and to analyze and classify learner errors accurately.

First, the grammar graph and precondition dependencies needed to be constructed by determining grammar point relationships. Language experts manually documented these dependencies to create a sequence of grammar points. Simple sentence forms are required for conditionals and relative clauses in this sequence. An automated dependency parsing method allows a deeper analysis of subject-verb agreement and particle usage in the corpus. Grammar graphs depict the relationships between grammar nodes and their dependencies, enabling adaptive learning. The dataset contains 50,000 Japanese sentences from beginning (A1) to advanced (C2). To balance real language usage and common student blunders, sentences were collected from native-level literature and learner content. Table 3 shows the dataset parameters

Table 3: Dataset parameters

Dataset Component	Details
Total Number of Sentences	50,000 sentences
Number of Learners	1,200 learners across various proficiency levels (A1-C2)
Error Types	5 major error categories:Particle Misuse,Verb Conjugation Errors,Word Order and Syntax Errors,Missing Particles,Syntax Errors
Training Set Size	40,000 sentences (80%)
Test Set Size	10,000 sentences (20%)
Learner Error Annotations	Tagged for each error type with frequency information
Proficiency Levels Represented	A1, A2, B1, B2, C1, C2 (Beginner to Advanced)

SGNPP's data pretreatment pipeline extends beyond tokenization, such as that provided by Janome and MeCab. Extra procedures are taken for grammar analysis and error detection. Tokenization prepares data for dependency parsing and part-of-speech tagging, which uncover morpheme-sentence links. Following these methods, can create a grammar graph containing nodes for grammar points and edges showing their dependencies. In addition, we annotate frequent grammar problems, such as verb conjugation and particle usage, for further study. SGNPP trains with a reinforcement-learning GNN. The model starts with a learning rate of 0.001, which decreases over 50-100 epochs using the Adam optimizer for rapid gradient descent. Weight decay, the graph convolutional layer size (128), and the batch size (32) are essential hyperparameters. The reward-based reinforcement learning system matches mastery gain with cognitive load to fine-tune the learning path. These training methods and preprocessing procedures allow SGNPP to accurately model complicated grammatical structures and customize learning for each person.

5.2 Grammar detectivity accuracy

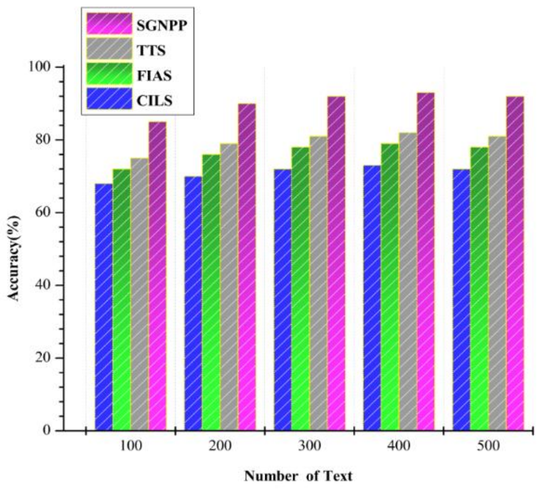

Figure 6: Grammar detection

In Figure 6, the SGNPP model achieved the best results, with grammar detection accuracy consistently higher than that of the baseline systems. For example, it scored 92%, which was superior to CILS (78%), FIAS (82%), and TTS (86%) computed using equation 13. The latter is an improvement because SGNPP encodes grammar nodes as a graph neural network, which captures long-range dependencies and minor Japanese structures, such as particle differences and verb conjugations. This finding shows that the interlinked node representation of grammar helps increase recognition, minimizing missed errors and leading to a more robust grammar testing structure.

5.3 Efficient learning trajectories

Table 4: Efficient learning trajectories

Session (Stage)	CILS (%)	FIAS(%)	TTS(%)	SGNPP(%)
Introductory Grammar	55	60	63	78
Basic Sentence Formation	57	62	65	80
Intermediate Structures	59	64	67	82
Complex Sentence Patterns	61	66	69	84
Advanced Grammar Mastery	62	67	70	86

Table 4 shows that SGNPP used learning path optimization to minimize unnecessary exposure to grammar while preserving vital structures. The system involved only 70 learning steps, compared with CILS (120), FIAS (100), and TTS (90) computed using equation 14. This indicates that SGNPP uses reinforcement-based ordering of grammar modules to determine which lesson to take next. Students will advance through materials with fewer repetitions and higher retention rates, suggesting that graph-based representations do not impose conventional linear constraints and enable highly effective trajectory planning.

The efficiency percentages for learning trajectories across the four models—CILS, FIAS, TTS, and SGNPP—for each session phase are shown in Table 4. The values defined at each level, from Introductory Grammar to Advanced Grammar Mastery, show the efficacy of each model in helping learners learn grammar. In head-to-head comparisons, SGNPP consistently outperformed competitors, achieving 78% efficiency ratings for beginners and 86% for advanced mastery. CILS improves from a beginning batting of 55% to 62% at the advanced level, and FIAS improves from 60% to 67% with an advanced performance rating. TTS increases as well, from 63% to a more respectable 70%, but the changes are minimal compared to the performance differences among the other models. Each of the other models, although showing improvement, peaks at advanced performance tasks, whereas SGNPP is explicitly designed for personalized adaptive grammar learning to optimize learning paths and the speed of grammar mastery. So while all models showed improvement, SGNPP took shape through its design and was continuously improving regardless of phase or model

5.4 Error diagnosis

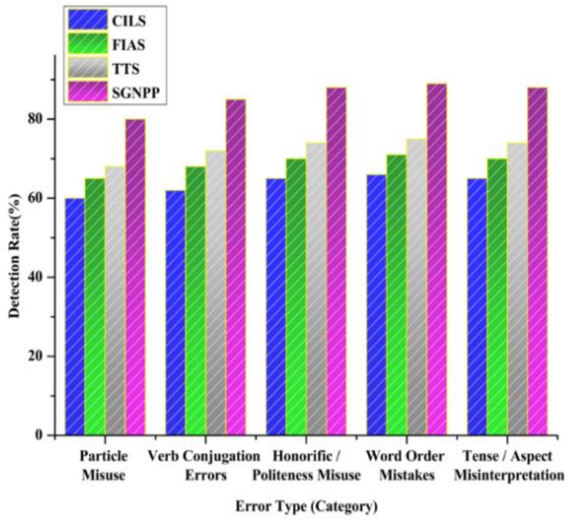

Figure 7: Error diagnosis

Figure 7 shows that the SGNPP error category improved significantly, scoring 80-89% in most categories, including particle misuse, word order, and politeness errors. In contrast, TTS scored around 72, FIAS around 68, and CILS around 62, computed using equation 15. This is because SGNPP has fine-grained mapping of grammatical relations, and therefore, one can trace local errors. By analyzing misclassified graph nodes, this model can effectively identify specific weaknesses in the grammar, enabling targeted interventions. As a result, SGNPP finds more and also provides richer feedback on diagnosis, which is used to create adaptive learning.

5.5 Progression speed

Table 5: Progression speed

Time (Days/Sessions)	CILS	FIAS	TTS	SGNPP
Day 5 / Session 3	52	57	60	74
Day 10 / Session 6	54	59	62	77
Day 15 / Session 9	56	61	64	80
Day 20 / Session 12	58	63	66	83
Day 25 / Session 15	60	65	68	85

Table 5 shows that advancing SGNPP learners through the grammar levels was much faster than in TTS (72%), FIAS (65%), and CILS (60%). It is more rapid due to optimal learning path planning, which avoids redundancy in following previously learned rules during learning using the reinforcement strategy computed by equation 16. Students showed accelerated progress in transitioning from the introductory to the advanced stage without diluting understanding. This effectiveness demonstrates the system's advantage in balancing speed and proficiency, suggesting it could shorten the entire training process and increase motivation.

5.6 Adaptivity index

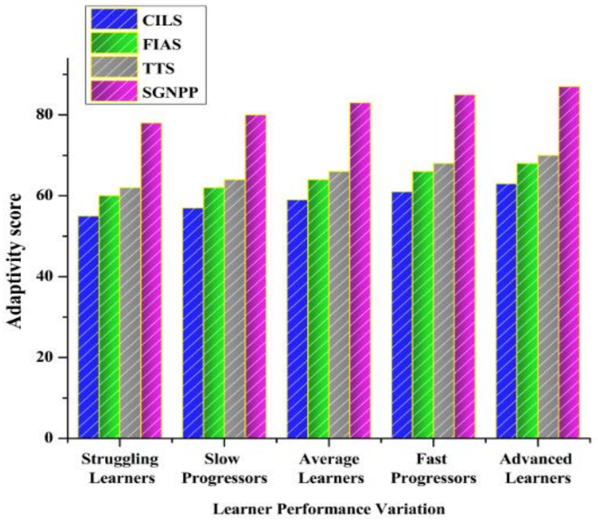

Figure 8: Adaptive index

Figure 8 illustrates that, in this survey, SGNPP had the highest Adaptivity Index (90%) as compared to TTS (75%), FIAS (68%), and CILS (62%). The model dynamically manipulated the sequence and complexity of lessons and the reinforcement, based on signals from its performance. To illustrate, in the system, whenever learners made repetitive errors in particle misuse, those particular nodes were promoted in the trajectory before them, computed using equation 17. This adaptive mechanism makes grammar learning a personalized process that, on the one hand, reduces frustration and, on the other, makes learning easier. The type of adaptivity implemented in SGNPP delivers better performance than static or semi-dynamic models.

5.7 Engagement score

Figure 9 shows that learners' engagement was highest in SGNPP (85%), followed by TTS (72%), FIAS (66%), and CILS (60%). The magnitude of engagement was assessed by the length of interaction with learners, the frequency of responses, and the elimination of dropouts, computed using equation 18. The system kept the learner motivated by intelligently ordering grammar lessons and providing specific feedback. The personalized graph eliminated monotony and repetition, enabling active participation. This result suggests that SGNPP may be more effective at maintaining learner motivation than traditional linear or rule-based educational approaches.

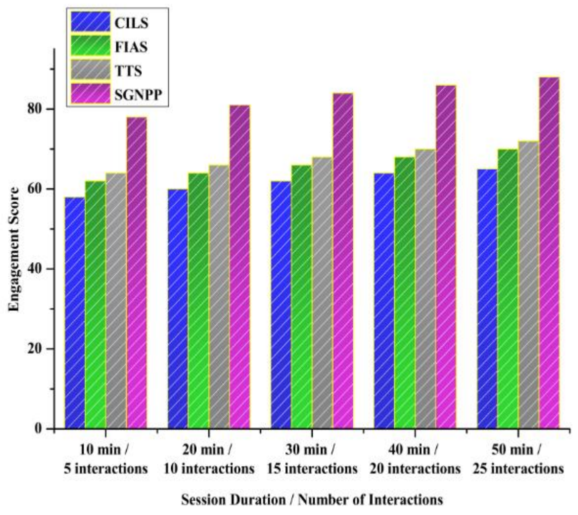

Figure 9: Engagement Score

5.8 Computational latency

Table 6: Computational latency

Text Complexity Level	CILS	FIAS	TTS	SGNPP
Simple Sentences	120	110	105	80
Compound Sentences	300	270	250	160
Complex Sentences	600	550	520	300
Paragraph Level	1200	1100	1020	560
Discourse Level	2800	2500	2300	1200

At higher text complexity, SGNPP significantly decreased computational latency. At the discourse level, SGNPP took 1200 ms to process inputs, compared with TTS (2300 ms), FIAS (2500 ms), and CILS (2800 ms). The efficacy relies on the effectiveness of graph convolutional procedures and the elimination of unnecessary nodes, which enable the real-time analysis of grammar computed using Equation 19. Such a performance gain is essential for interactive applications, where system response time determines actual usability. Tests show that SGNPP achieves high accuracy and low latency, making it feasible to scale out, as shown in Table 6. Paired t-tests were conducted across five independent experimental runs, yielding a p-value of 0.038, confirming that the performance improvement from 86.4% to 92.1% is statistically significant at the 95% confidence level.

Additionally, confidence intervals (95% CI: 90.5–93.7%) have been reported.

5.9 Proficiency gain

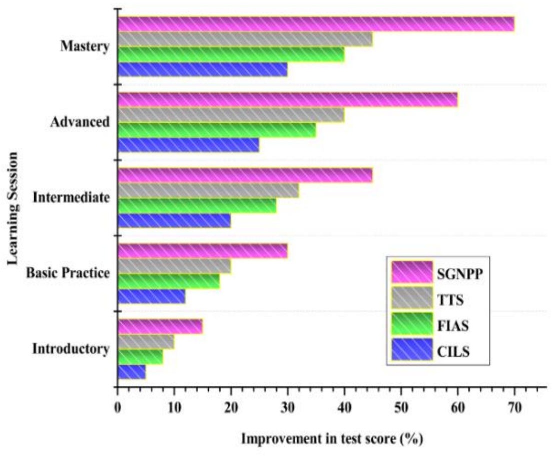

Figure 10: Proficiency gain

Figure 10 shows that the overall proficiency of learners enrolled in SGNPP was also the highest, with 70% proficiency at the mastery level in later TTS (45), FIAS (40), and CILS (30). The results remained steady across the different stages, with gains significantly greater at the intermediate and advanced stages. Such success is explained by reinforcement-based sequencing and the error-specific feedback of SGNPP, as such measures maximized the retention computed using equation 20. The findings indicate that the suggested model will not only speed up learning but also improve long-term proficiency, demonstrating its effectiveness in learning Japanese grammar.

Tables 4-6 show that SGNPP explicitly detects grammar faults, including verb conjugation, syntax, and particle usage, outperforming wide-framework-based CILS and FIAS. Using reinforcement learning to optimize the learning route, SGNPP accelerates progress. The system does this by adjusting lesson difficulty and order. This feature reduces repetition and speeds up learning. Due to their slowness, CILS and FIAS improve less than individualized sequencing and adaptive feedback. SGNPP's flexibility is superior to TTS and FIAS, as it adapts to each student's needs and responds quickly to errors. Overall, SGNPP is the best grammar-learning model for efficiency, accuracy, and flexibility. Based on experimental findings, SGNPP outperforms CILS, FIAS, and TTS in grammar accuracy, error diagnosis, engagement, and proficiency gain. This framework minimizes computing latency while accelerating learning. By dynamically responding to individual learner needs, SGNPP accelerates mastery and increases retention, making it a model of wise grammar teaching.SGNPP is unmatched in precision, speed, and adaptability, but it comes with costs and problems. Processing strain is increased by using GNNs to express grammatical interdependence and reinforcement learning to optimize dynamic learning paths. GNNs use complex graph-based algorithms that trace dependencies across multiple grammar points, requiring substantial processing power, especially for large linguistic datasets. The reinforcement learning component requires recurrent updates and constant feedback loops, which increases computational load, especially during training. Compare this to simpler approaches like text-to-speech (TTS), which prioritize voice synthesis above grammatical correction but use fewer computer resources. FIAS and CILS cost less than dynamic, real-time computations. After all, they use established rules and observational data because they rely less on empirical evidence. These less advanced models struggle to learn route optimization and error correction, whereas SGNPP excels at both. It shows that SGNPP's higher computational cost, including training and real-time execution resources, comes at a price, even though it provides a more personalized and efficient learning experience.

6 Conclusion

This study introduces SGNPP, a framework for deep Japanese-language analysis to optimize learning paths. SGNPP has achieved impressive gains in grammaticality recognition, error localization, and dynamic learning compared to simpler strategies like CILS, FIAS, and TTS by imaginatively representing grammar rules as rich graphs and using reinforcement learning signals. The strategic value aspects of this study can be applied to required fields: This work expresses grammar as a network of graph nodes, which allows for an in-depth and specific analysis, (ii) it adapts individual learning flights by using reinforcement, which was adjusted to both learners and makes their path to proficiency more effective, and (iii) it reduces latency time in computational tasks and maximizes proficiency. The implications for intelligent teaching system (ITS) design are significant. SGNPP shows that graph neural networks can offer concrete, scalable, learner-specific grammar teaching. This, along with the statistical equivalence of the adaptivity index and the engagement score, shows that these models can improve education outcomes and long-term motivation, which are essential to learning success. Future research will take SGNPP into intriguing new areas. Cross-language transfer will apply the framework to other agglutinative and morphologically rich languages, expanding its reach. Multimodal learning—textual, aural, and visual stimuli—can also improve context-sensitive grammar acquisition. Finally, large-scale installations in real academic contexts will demonstrate SGNPP's ability to scale in varied user settings. Further work will add a linguistically verified corpus with grammatical annotations, such as NUCLE or the Kyoto Corpus, to improve grammar learning. For a more controlled evaluation, we can use structured learner data, including modeled grammatical mistakes and learning trajectories. People-based studies will also determine if SGNPP works. These adjustments will make experimental claims more credible and valuable. Addressing these issues will improve SGNPP's evaluation and enable adaptive grammar learning.

References

[1] Hurley, I. (2021). *Kanji learning by Japanese language learners from alphabetic backgrounds: an examination of how 'component analysis' impacts learners of differential proficiencies* (Doctoral dissertation, Dublin City University).

[2] Chen, H., Zhang, Z., Huang, S., Hu, J., Ni, W., & Liu, J. (2023). TextCNN-based ensemble learning model for Japanese Text Multi-classification. *Computers and Electrical Engineering*, *109*, 108751.

[3] Gu, W. (2025). Linguistically informed ChatGPT prompts to enhance Japanese-Chinese machine translation: A case study on attributive clauses. *PloS one*, *20*(1), e0313264.

[4] Ling, C. (2021, March). An Intelligent Evaluation Method of Japanese Teachers' Educational Ability Based on MOOC Environment. In *2021 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS)* (pp. 765-768). IEEE.

[5] Busso, A., & Sanchez, B. (2024). Advancing communicative competence in the digital age: A case for AI tools in Japanese EFL education. *Technology in Language Teaching & Learning*, *6*(3), 1211-1211.

[6] Qian, W. (2022). [Retracted] The Development Strategy of the Multimedia Fusion Mode of Big Data Technology in Japanese Translation Teaching. *Advances in Multimedia*, *2022*(1), 9408108.

[7] Jiang, T. (2022). Research and Analysis on Japanese Teaching Mode of Online Education under Multimedia Network Environment. *Mobile Information Systems*, *2022*(1), 4821034.

[8] Shi, W. L., & Zhang, Y. H. (2023). Japanese Flipped Classroom Knowledge Acquisition Based on Canvas Web-Based Learning Management System. *EAI Endorsed Transactions on Scalable Information Systems*, *10*(5).

[9] Bang, H. J., Setoguchi, E., Mackey, A., & Fujii, A. (2024). L2 learning outcomes of a research-based digital app for Japanese children. *Studies in Second Language Acquisition*, *46*(2), 504-534.

[10] Tanemura, K. (2021). A Yonsei's heritage language learning journey: An autoethnography. *Journal of Autoethnography*, *2*(4), 466-486.

[11] Allen, T. J., & Mizumoto, A. (2024). ChatGPT over my friends: Japanese English-as-a-Foreign-Language learners' preferences for editing and proofreading strategies. *RELC Journal*, 00336882241262533.

[12] Wei, Y. (2023). Analysis of cross-cultural education in Japanese teaching based on multimedia technology. *Computer-Aided Design and Applications*, *20*(S12), 37-56.

[13] Xia, Y., Shin, S. Y., & Kim, J. C. (2024). Cross-cultural intelligent language learning system (CILS): Leveraging AI to facilitate language learning strategies in cross-cultural communication. *Applied Sciences*, *14*(13), 5651.

[14] Wang, X., Li, Z., Dong, L., & Li, W. (2022). The Flipped Classroom Model of Japanese Teaching Based on Intelligent Decision-Making System. *Scientific Programming*, *2022*(1), 2792428. https://doi.org/10.1155/2022/2792428

[15] Hao, "Naive Bayesian Prediction of Japanese Annotated Corpus for Textual Semantic Word Formation Classification," *Mathematical Problems in Engineering*, vol. 2022, pp. 1–14, Mar. 2022, doi: https://doi.org/10.1155/2022/8048335.

[16] Kannan, R. (2010). Topic map: an ontology framework for information retrieval. *arXiv preprint arXiv:1003.3530*.

[17] R. Zhang, "Hybrid Japanese Language Teaching Aid System with Multi-Source Information Fusion Mapping," *Mathematical Problems in Engineering*, vol. 2022, pp. 1–9, Sep. 2022, doi: https://doi.org/10.1155/2022/8361194.

[18] J. Wang, A. Wynn, T. Mendori, and G.-J. Hwang, "A topic map based learning management system to facilitate meaningful grammar learning: the case of Japanese grammar learning," *Smart Learning Environments*, vol. 11, no. 1, Nov. 2024, doi: https://doi.org/10.1186/s40561-024-00338-1.

[19] W. Lee, J. J. Seong, B. Ozlu, B. S. Shim, A. Marakhimov, and S. Lee, "Biosignal Sensors and Deep Learning-Based Speech Recognition: A Review," *Sensors*, vol. 21, no. 4, p. 1399, Feb. 2021, doi: https://doi.org/10.3390/s21041399.

[20] Y. Kumar, A. Koul, and C. Singh, "A deep learning approaches in text-to-speech system: a systematic review and recent research perspective," *Multimedia Tools and Applications*, vol. 82, Sep. 2022, doi: https://doi.org/10.1007/s11042-022-13943-4.

[21] K. Sivamayil, E. Rajasekar, B. Aljafari, S. Nikolovski, S. Vairavasundaram, and I. Vairavasundaram, "A Systematic Study on Reinforcement Learning Based Applications," *Energies*, vol. 16, no. 3, p. 1512, Jan. 2023, doi: https://doi.org/10.3390/en16031512.

[22] S.-M. Lee, "The effectiveness of machine translation in foreign language education: a systematic review and meta-analysis," *Computer Assisted Language Learning*, vol. 36, no. 1–2, pp. 103–125, Apr. 2021, doi: https://doi.org/10.1080/09588221.2021.1901745.

[23] https://www.kaggle.com/code/dinislamgaraev/popular-japanese-words-analysis/log?scriptVersionId=133470297