Neural Wani: Toward Accelerating the Automated Theorem Prover *wani* for Dependent Type Theory

Nanako Miyagawa and Hinari Daido* **and Daisuke Bekki**
Ochanomizu University
{miyagawa.nanako, daido.hinari, bekki}@is.ocha.ac.jp

Abstract

This paper proposes *Neural Wani*, an integration of a neural model into the automated theorem prover *wani* for Dependent Type Theory (DTT), aimed at accelerating proof search in natural language inference (NLI) pipelines. We implemented a lightweight LSTM-based model to predict the probability distribution of applicable inference rules and integrated it into *wani*'s backward inference process. Evaluation using the JSeM dataset demonstrates that *Neural Wani* achieves a 1.41× speedup compared to the standard non-neural baseline. Although slight overhead is observed in simpler proofs, our results indicate that neuro-symbolic integration effectively guides search in complex DTT-based automated theorem proving.

1 Introduction

Elucidating the computational processes underlying the structural meaning of natural language is a foundational goal in computational linguistics. While recent data-driven methods have shown remarkable performance, pipeline systems based on formal linguistics remain crucial for achieving strict interpretability and verifiable reasoning in natural language inference (NLI). Throughout this paper, NLI denotes the task of deciding, for a set of premises and a candidate conclusion, whether the premises entail the conclusion, contradict it, or do neither. Because clarifying these linguistic processes through manual derivation is prohibitively complex, establishing robust, automated NLI pipelines is in high demand.

To this end, Tomita et al. (2025) developed a comprehensive NLI pipeline integrating *lightblue* (Bekki and Kawazoe, 2016)[1]—a CCG-based syntactic and semantic parser—with *wani* (Daido and Bekki, 2020), an automated theorem prover designed for Dependent Type Theory (DTT) (Martin-Löf, 1984). This system processes raw text through syntactic and semantic analysis, type checking, anaphora and presupposition resolution, and automated theorem proving.

A key strength of this architecture is its highly transparent syntax-semantics interface, afforded by CCG (Steedman, 1996), which compositionally maps surface strings into complex DTS semantic representations. Despite this structural elegance, the practical deployment of such a pipeline is severely constrained by the computational cost of the final phase: automated theorem proving. The vast search space inherent to DTT makes purely logic-based proof search inefficient, necessitating a more guided approach.

1.1 *wani*

Wani, the automated theorem prover integrated into the aforementioned NLI system, adopts DTT as its proof system. Concretely, *wani* is based on a natural-deduction-style formulation of DTT, in which each type former is governed by formation (F), introduction (I), and elimination (E) rules, and hypotheses may be introduced and later discharged within a derivation. This follows the standard natural-deduction presentation of Martin-Löf's intuitionistic type theory (Martin-Löf, 1984), on which DTT is based. A judgment in DTT is expressed in the following form:

$$\Gamma \vdash M : A$$

where Γ represents the context, M the term, and A the type. From the perspective of type theory, this judgment signifies that "the term M has type A under the context Γ." Simultaneously, from the viewpoint of proof theory, it indicates that "there exists a proof M for the conclusion A given the premises Γ" via the Curry-Howard correspondence.

*This work was conducted independently and does not reflect the views or positions of Amazon.

[1]https://github.com/DaisukeBekki/lightblue/

Third Workshop on the Bridges and Gaps between Formal and Computational Linguistics, pages 12–21

A judgment in which the proof term M is unknown, as shown below, is called a *proof search query*, and the task of finding a proof term for such a query is referred to as *proof search*.

$$\Gamma \vdash ? : A$$

To resolve a proof search query, *wani* employs a combination of forward and backward inference. Forward inference focuses on "what can be derived from the premises," constructing a proof tree from the top down. Conversely, backward inference focuses on "what is required to prove a given proposition," constructing the proof tree from the bottom up. In this process, inference rules are applied to the target proposition to identify the necessary conditions as subgoals.

1.2 Dependent Type Semantics (DTS)

The natural language inference system described in Section 1 adopts Dependent Type Semantics (DTS) (Bekki, 2014; Bekki and Mineshima, 2017) as its framework for semantic representation and composition. DTS is a theory of natural language semantics based on DTT. In DTT, Π-types and Σ-types are used as generalizations of function types and product types, denoted as $(x : A) \to B$ and $(x : A) \times B$, respectively. Since the type of the consequent B can depend on the proof of the antecedent A, DTS is capable of representing sentence meanings that depend on the meanings of preceding sentences. This property allows DTS to handle complex linguistic phenomena such as anaphora resolution and presupposition binding.

Furthermore, DTS is a form of proof-theoretic semantics, which defines the meaning of a sentence as its verification condition mediated by inference rules. Compared to model-theoretic semantics—the standard theory in formal semantics—DTS offers the implementation advantage of computing the success or failure of inference solely through proof search, without the need to refer to external models. However, DTT—the higher-order foundation of DTS—possesses greater expressive power than first-order logic but faces computational challenges such as undecidability. To address this, we build upon the framework of *Neural Wani* (Miyagawa et al., 2025), introducing a neuro-symbolic integration to reduce *wani*'s computation time by optimizing proof search efficiency.

2 Background

Neural Wani is an initiative aimed at accelerating inference by integrating neural models into the proof search process of DTT. The traditional backward inference in *wani* relies on depth-first search (DFS) with predefined depth and time limits. This approach faces a significant challenge: computation time increases exponentially if inappropriate inference rules are applied during the early stages of the search. To address this, *Neural Wani* employs a method that takes a DTT judgment as input and uses a neural model to predict the most applicable inference rule. By prioritizing the search paths with higher predicted probabilities, the system substantially reduces the overall time required for proof search.

In a preliminary study (Miyagawa et al., 2025), various vector embedding methods for DTT judgments were compared and evaluated. Based on those evaluation results, an embedding strategy specifically optimized for the structural requirements of DTT proof search was established, serving as the foundational representation for the predictive model in this work.

Although *Neural Wani* originates from this preliminary study (Miyagawa et al., 2025), the present work extends it in three respects. **(i) Task setting:** the prior model predicted inference rules from judgments that still contained proof terms, whereas we address the strictly harder and practically essential setting of predicting rules directly from *proof search queries*—judgments that lack proof terms (Section 3.1). **(ii) Integration:** rather than evaluating the predictor in isolation, we embed it into *wani*'s actual backward inference loop, so that the predicted rule ordering directly governs the proof search. **(iii) Evaluation:** we measure the resulting end-to-end speedup on an NLI pipeline using held-out JSeM problems, instead of reporting classification accuracy alone (Section 4.3).

3 Proposed Method

3.1 Predicting Rules from Proof Search Queries

Previous work (Miyagawa et al., 2025) developed a model to predict inference rules using judgments that already contained proof terms as input. In DTT, a proof term explicitly encodes the application history of inference rules; consequently, predicting rules from such judgments is relatively straightforward, as evidenced by a reported high micro-F1

score of 0.982. However, during the actual process of proof search (backward inference), as discussed in Section 1.1, the proof term is unknown. A practical automated prover must therefore be capable of predicting inference rules directly from proof search queries, which are judgments lacking proof terms.

In backward inference, applying an inference rule to a current judgment generates new subgoals, which are themselves judgments without proof terms. By recursively predicting the appropriate rule names at each of these stages, the entire proof tree can be constructed. Therefore, the task addressed in this study—predicting rules under the condition where the proof term is missing—marks a crucial step toward achieving effective automated search in *Neural Wani*.

To make the setting concrete, consider a toy NLI problem. We use a Japanese example, in keeping with our JSeM-based evaluation, transliterated in romaji with English glosses in parentheses: from the premises "Subete no gakusei ga aruita" ("Every student walked") and "John wa gakusei de aru" ("John is a student"), the hypothesis "John wa aruita" ("John walked") follows. For readability we use a simplified DTS representation in which $\mathbf{student}$ and $\mathbf{walk}$ are treated as unary predicates over entities; the representations that *lightblue* actually produces additionally encode event variables, tense, and multi-place predicate–argument structure, which we abstract away here.[2] Over a signature declaring $\mathbf{student}, \mathbf{walk} : \mathsf{entity} \to \mathsf{type}$ and $\mathbf{john} : \mathsf{entity}$, the premises correspond to a context Γ, and the prover attempts the query $\Gamma \vdash ? : \mathbf{walk}(\mathbf{john})$. Figure 1 shows the resulting proof, which *wani* builds by backward inference using only (ΠE) and $(\mathrm{Membership})$; since a proof term is found, the verdict is YES (entailment).

3.2 Tokenization and De-lexicalization

To handle the open vocabulary of natural language and focus on the underlying logical structure of DTT judgments, we implemented a de-lexicalization strategy. Instead of relying on raw surface strings or high-dimensional word embeddings, we map each element of a judgment to a predefined set of abstract tokens:

- **Structural Tokens**: Logical constants (e.g., Π, Σ, $=$, `type`) and delimiters (e.g., `EOCon`, `EOSig`) are explicitly assigned unique tokens to preserve the formal syntax of DTT.

- **Constant Placeholders**: Lexical constants appearing in a judgment are sequentially mapped to generic tokens `Word1` through `Word31` based on their order of appearance. This threshold was empirically determined to cover the majority of our dataset; any constants exceeding this limit are mapped to an `UNKNOWN` token.

- **Variable Placeholders**: Bound variables are similarly mapped to `Var'0` through `Var'6` based on De Bruijn indices, with any overflow mapped to `Var'unknown`. This mapping naturally resolves α-equivalence issues, allowing the model to treat structurally identical terms with different variable names as the same sequence.

This approach ensures that the model remains agnostic to specific lexical semantics, compelling it to learn the abstract structural templates that drive proof search. Because lexical content is abstracted away in this manner, the predictor depends solely on the structural form of DTT judgments. It is therefore independent of the surface language analyzed upstream by *lightblue*, and, although our experiments target NLI, the same mechanism applies to any proof search query expressible in DTT.

3.3 Embedding Representation of Judgments

In this study, we employ the "EO~delimiter" method to encode proof search queries for the neural model. In a previous evaluation of embedding strategies (Miyagawa et al., 2025), comparing four embedding methods—parentheses delimiters, SEP delimiters, EO~delimiters, and no delimiters—the EO~delimiter method achieved the highest micro-F1 score. Furthermore, because the current task is strictly more complex due to the absence of proof terms, explicitly demarcating the logical roles and boundaries of each element within the judgment is essential. Therefore, the EO~delimiter representation is particularly well suited to this setting.

The specific data structure is as follows. Because the proof term is inherently unknown during the proof search, the `term` component is excluded from the input representation. For the goal judgment

[2] For instance, *lightblue* renders "John walked" as $(x{:}\mathsf{entity}) \times (\mathbf{walk}(x, \mathbf{john}, @\mathsf{entity})) \times (\mathbf{Past}(x))$ rather than the simplified $\mathbf{walk}(\mathbf{john})$ used here.

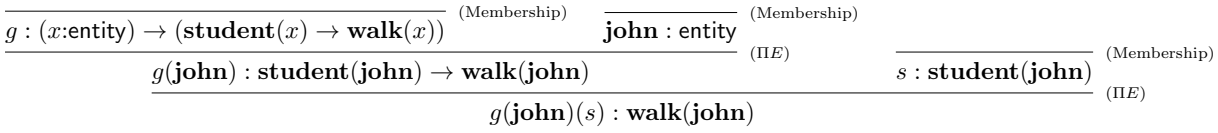

Figure 1: A toy NLI proof built by *wani*: from "Subete no gakusei ga aruita" ("Every student walked") and "John wa gakusei de aru" ("John is a student"), it derives "John wa aruita" ("John walked"; $\mathbf{walk}(\mathbf{john})$) by backward inference. The representation is simplified for illustration (see the main text). Leaves labeled (Membership) retrieve premises from the context and constants from the signature.

$\Gamma \vdash ? : \mathbf{walk}(\mathbf{john})$ of the toy proof in Figure 1, the components are:

```
signature = [(student, entity->type),
(walk,     entity->type),     (john,
entity)]

context = [student(john),
(x:entity)->(student(x)->walk(x))]

~~term = ...~~    ← Excluded in this study

type = walk(john)
```

After de-lexicalization, which replaces the constants `student`, `walk`, and `john` with `Word1`, `Word2`, and `Word3` (and the bound variable x with `Var'0`/`Var'1` according to its De Bruijn index), this judgment is encoded as the following EO~delimiter token sequence (grouped by component):

```
Word1 COMMA Pi' Entity' EOPre Type' EOPre
EOPre EOPair
Word2 COMMA Pi' Entity' EOPre Type' EOPre
EOPre EOPair
Word3 COMMA Entity' EOPre EOPair EOSig
App' Word1 EOPre Word3 EOPre EOPre
Pi' Entity' EOPre Pi' App' Word1 EOPre
Var'0 EOPre EOPre App' Word2 EOPre Var'1
EOPre EOPre EOPre EOPre EOCon
App' Word2 EOPre Word3 EOPre EOPre EOTyp
```

3.4 Neural Model Architecture

Following previous work (Miyagawa et al., 2025), we constructed the rule prediction model using Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997). The specific hyperparameters used in our experiments are detailed in Table 1.

Table 1: Hyperparameters

Number of Epochs	10
Learning Rate	5×10^{-4}
Number of Layers	1
Dropout	None
Input Size	128
Hidden State Size	128
BPTT Steps	32

While Transformer-based architectures (Vaswani et al., 2017) have become the standard in various sequence-to-sequence tasks, we deliberately opted for a lightweight LSTM for our rule prediction task. The primary rationale lies in the strict latency constraints of automated theorem proving. In backward inference, the model is queried recursively at every node of the expansive search tree. The heavy multi-head attention mechanisms of Transformers introduce significant computational overhead, which could easily negate the speedup gained from pruning the search space.

Furthermore, unlike general natural language processing, where long-range global context or external discourse information may alter the interpretation, the "context" required for rule prediction is entirely self-contained within the localized structure of the judgment itself. Given this structural locality, the sequential processing capability of a lightweight LSTM suffices, balancing structural pattern recognition against the low-latency inference required to accelerate the prover.

4 Evaluation Experiments

4.1 Training and Evaluation Dataset

We used Natural Language Inference (NLI) problems collected from JSeM (Kawazoe et al., 2015a,b), a Japanese semantic test suite, as our dataset. For each problem, we performed CCG-based syntactic parsing using the *lightblue* parser, followed by semantic composition within the DTS framework. The resulting type-checked proof diagrams serve as the primary data for this study. We selected eight specific sections to ensure coverage across a broad range of linguistic phenomena, including generalized quantifiers, verbs, adjectives, compound adjectives, propositional attitudes, nominal anaphora, noun phrases, and temporal reference.

In this experiment, to prevent data leakage across identical proof contexts, we allocated 90% of the

676 valid JSeM problems for model training, validation, and testing, strictly reserving the remaining 10% for a downstream comparative performance analysis between *Neural Wani* and the baseline *wani* system. From the allocated 90% problem set, we extracted the training data according to the following procedure:

1. We performed type checking with *wani* to generate proof trees.
2. We extracted pairs consisting of a judgment and its corresponding applied inference rule name from the resulting proof trees.
3. From the extracted pairs, we selected only the rules used in *wani*'s backward inference, explicitly excluding pairs corresponding to formation rules.
4. To construct the final dataset, we collected up to 2,000 data points per selected rule. For rules with fewer than 2,000 available instances, we used all available data.

The capping in Step 4 is crucial for addressing the severe class imbalance inherent in proof search; majority rules such as $(\mathrm{Membership})$[3] and (ΠE) occur significantly more often than others. By limiting these dominant classes, we mitigated model bias and ensured more robust generalization across the rules that occur in the data; Figure 2 contrasts the distribution before and after capping. Although Table 2 lists ten backward-inference rules, the training data are drawn only from non-formation rules (Step 3 above excludes formation rules), and among these, only (ΠI), (ΠE), $(\mathrm{Membership})$, and (ΣI) actually occur in the proof trees extracted from our eight JSeM sections. Rules such as $(\top I)$, (DNE), and (EFQ) do not appear in this data and are therefore absent from both the plot and the per-label results in Table 4. Among the four observed rules, (ΣI) is extremely rare (only two instances), so in practice the model learns to distinguish the three remaining rules.

Through this controlled procedure, we obtained a total of $4,689$ judgment–rule pairs. We partitioned this final dataset into training (80%), validation (10%), and test (10%) sets, using a stratified split so that the per-rule label proportions are identical across all three subsets.

[3] $(\mathrm{Membership})$ is not a primitive rule of DTT but an implementation-level rule of *wani*: it proves a judgment by looking up a matching declaration, covering two cases—retrieving a constant from the signature, which corresponds to DTT's (CON) rule, and retrieving an assumption from the context, which is not given a named rule in standard DTT formulations.

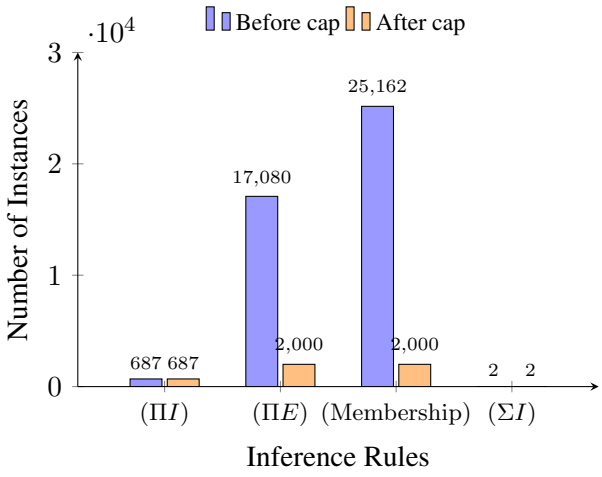

Figure 2: Label distribution over the four backward-inference rules that occur in the dataset, shown before and after the 2,000-instance cap. Only (ΠE) and $(\mathrm{Membership})$ exceed the cap, whereas (ΠI) and (ΣI) remain unchanged.

4.1.1 Rationale for Focusing on Backward Inference Rules

The primary objective of *Neural Wani* is to use neural models to improve the efficiency of computationally expensive stages within the proof search process. In *wani*'s inference system, the rules applicable to forward reasoning are strictly limited, whereas the candidate rules applicable at each step of backward inference are far more diverse, as shown in Table 2. Furthermore, certain rules—such as (ΠE) and (DNE)—can be applied to any type without restriction, which frequently leads to an explosive expansion of the search space. Because these backward inference steps inherently involve a vast search space and represent the primary computational bottleneck, this study focuses specifically on predicting the inference rules used during backward inference.

Table 2: Inference rules in *wani*, grouped by direction. $(\mathrm{Membership})$ appears in both the forward and backward sets.

Forward	**Backward**		
$(\mathrm{Membership})$	(ΠI)	(ΣI)	$(\mathrm{Membership})$
(ΣE)	(ΠE)	(ΣF)	(DNE)
$(=I)$	(ΠF)	$(\top I)$	(EFQ)
$(=E)$	$(=F)$		

4.1.2 Rationale for Excluding Formation Rules

Formation rules in DTT are represented as follows:

$$\frac{\begin{array}{c}\overline{x : A}^{\,i} \\ \vdots \\ A : \mathsf{s}_1 \qquad B : \mathsf{s}_2\end{array}}{(x : A) \to B : \mathsf{s}_2}(\Pi F), i$$

$$\text{where } (\mathsf{s}_1, \mathsf{s}_2) \in \{(\mathsf{type}, \mathsf{type}), (\mathsf{type}, \mathsf{kind}), (\mathsf{kind}, \mathsf{kind})\}$$

Figure 3: Example of the (ΠF) rule

Following the natural-deduction convention, the overlined assumption $\overline{x : A}^{\,i}$ in Figure 3 denotes a hypothesis that is discharged at the inference step labeled i.

$$\frac{A : \mathsf{type} \quad M : A \quad N : A}{M =_A N : \mathsf{type}}(=F)$$

Figure 4: Example of the $(=F)$ rule

In DTT, the conclusion (the lower part) of a formation rule typically yields a judgment with either type or kind as its type. When a judgment possesses these specific types, the applicable rule is syntactically and uniquely determined, with few exceptions such as (CON). In other words, the application of formation rules is deterministic and does not necessitate the probabilistic prediction provided by a neural model. For this reason, we explicitly excluded formation rules from the scope of our prediction task and evaluation experiments.

4.2 Evaluation Methodology for *Neural Wani*

We integrated the rule prediction model into the baseline *wani* system to investigate whether it effectively improves the search.[4] For this evaluation, we randomly selected 50 NLI proof search problems from the reserved data strictly excluded from the training, validation, and testing sets.

The evaluation process for *wani* was conducted according to the format illustrated in Figure 5. Specifically, *wani* returns YES—meaning that the premises entail the conclusion—if a proof term for the problem $\Gamma \vdash ? : A$ is discovered, and NO—meaning that the premises contradict the conclusion—if a proof term for the negation $\Gamma \vdash ? : \neg A$ is found instead. If the system fails to find either within the specified time limit,[5] it returns UNK (Unknown), the case in which neither entailment nor contradiction can be established. For this evaluation, *wani* was configured to operate in intuitionistic mode with a search depth limit of 9. *wani* provides two mutually exclusive modes—an intuitionistic mode that uses (EFQ) and a classical mode that uses (DNE)—so the two rules are never available simultaneously; accordingly, (DNE) is not used in our experiments.

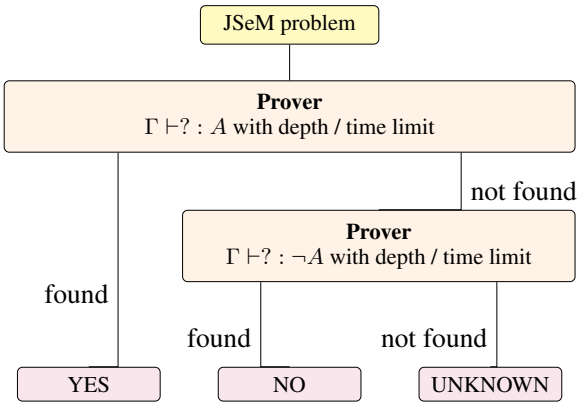

Figure 5: Evaluation Method

Experimental precautions. To ensure a fair comparison and minimize experimental bias, several technical precautions were implemented. First, to account for lazy evaluation in Haskell (the implementation language of *wani*), we ensured that proof search queries were fully evaluated and manual garbage collection was triggered before each test case to prevent resource interference between runs. Second, to mitigate potential system-level execution biases, we alternated the execution order: for half of the test problems, the baseline *wani* was executed first, while for the other half, *Neural Wani* was executed first. Finally, while *Neural Wani* employs a caching mechanism to store rule predictions for recurring judgments within a single proof search, this cache was strictly cleared between experiments under different time limits to ensure the independence of each measurement.

[4] During the search process, formation rules (which were excluded from the training data) are applied first. Subsequently, *Neural Wani* predicts the priority of backward inference rules, and the search proceeds according to this predicted order. To fully assess the model's predictive capability, we did not perform top-k pruning; rather, all applicable rules were considered in the sorted order.

[5] In *wani*, we impose depth and time limits on the deduce function, which performs a single backward inference.

4.3 Experimental Results

4.3.1 Model Evaluation

Table 3: Overall classification performance

	Prec	Rec	F1	Supp
micro	0.821	0.821	0.821	470
macro	0.834	0.608	0.591	470
w-avg	0.834	0.821	0.821	470

Table 4: Classification performance per label

	Prec	Rec	F1	Supp
(ΠI)	0.582	0.768	0.662	69
(ΠE)	0.866	0.710	0.780	200
(Membership)	0.888	0.955	0.920	200
(ΣI)	1.000	0.000	0.000	1

4.3.2 Performance Comparison between Normal and Neural Approaches

Table 5: Performance comparison between the baseline *wani* (Normal) and *Neural Wani* (Neural) across time limits. Out of 50 NLI problems, both systems solved 8 under every limit, so the success rate is identical and we report only timing. "all" averages over all 50 problems (including timeouts), whereas "both succ." averages only over problems solved by both systems. Speedup is the ratio of Normal to Neural time.

Time limit	Avg. time	Normal	Neural	Speedup
30s	all (sec)	180.619	128.376	1.41×
	both succ. (sec)	2.093	4.915	0.43×
60s	all (sec)	309.516	219.143	1.41×
	both succ. (sec)	2.068	2.087	0.99×
90s	all (sec)	404.035	314.057	1.29×
	both succ. (sec)	2.062	2.082	0.99×

5 Discussion

5.1 Neural Model

As shown in Table 3, the overall evaluation yielded a high micro-F1 score of 0.821, whereas the macro-F1 score remained at 0.591. This discrepancy stems from the imbalanced label distribution within the proof search dataset. Since the micro-F1 score reflects the overall accuracy across all instances, it is heavily influenced by the model's performance on majority classes. In contrast, the lower macro-F1 score—which treats all classes equally—indicates that predicting minority rules remains a challenge. We further note that this micro-F1 of 0.821 is lower than the 0.982 reported by the prior model that operated on judgments containing proof terms (Miyagawa et al., 2025). This gap is expected rather than a regression: our setting removes the proof term—the single most informative cue for identifying the applied rule—and therefore constitutes a substantially harder prediction problem, as discussed in Section 3.1.

However, from the perspective of our primary objective—accelerating the proof search pipeline with *Neural Wani*—this performance profile is highly advantageous for the following reason.

The robust micro-F1 score of 0.821 directly translates to overall search efficiency. In DTT proof search, the majority of judgments involve high-frequency rules such as (Membership) and (ΠE). Maintaining high predictive accuracy for these dominant rules ensures that, at most nodes in the search tree, the model correctly prioritizes the optimal inference rule. This successful pruning directly reduces the exponentially growing computation time of depth-first search.

In conclusion, while there is room for improvement in predicting long-tail rules, the proposed model demonstrates high practical performance as a supportive tool for accelerating the DTT prover *wani*.

5.2 Quantitative Analysis of Search Efficiency

As shown in Table 5, *Neural Wani* maintains exactly the same success rate as the non-neural baseline *wani*: both systems solved 8 out of 50 highly complex NLI problems under all time limits. This result confirms that the integration of the neural model safely preserves the logical completeness of the original prover; problems solvable by the baseline remain solvable with *Neural Wani*.

While the absolute success rate remains unchanged, *Neural Wani* demonstrates a consistent reduction in average execution time across all test cases, including those that ultimately resulted in timeouts. Notably, the overall average runtime achieved a speedup of up to 1.41× under the 30s and 60s limits. This marked acceleration suggests that the neural-guided prioritization effectively steers the prover away from exploring deep, unproductive branches in the search space.

Conversely, an analysis of problems that are solved rapidly by both systems reveals a well-known neuro-symbolic trade-off. In structurally shallow cases, *Neural Wani* occasionally incurs

slight overhead due to the constant inference time of the LSTM. This indicates that when the search space is shallow, the computational cost of neural evaluation may not be entirely offset by the benefit of heuristic pruning.

Taken together, these observations confirm that *Neural Wani* is well suited to its design goal: it substantially improves search efficiency in large, intractable search spaces where traditional provers struggle, while introducing only a modest overhead in already trivial cases.

5.3 Cognitive Parallelism: Intuition-Guided Search

Beyond the quantitative gains, the integration of a neural model into *wani*'s symbolic engine admits an interpretive analogy—we stress that it is an analogy, not a cognitive claim—through Dual Process Theory (Evans and Frankish, 2009): the LSTM classifier functions as **System 1**, providing rapid heuristic suggestions from the structural features of a judgment, while *wani* acts as **System 2**, verifying them through backtracking and type checking. What makes this synergy specific to *Neural Wani*—rather than generic to any neuro-symbolic prover—is the locus at which System 1 intervenes, namely the order in which inference rules are expanded during backward search; we contrast this locus with premise selection and differentiable proving in Section 6. These results are consistent with this interpretation: the predictor attains its highest accuracy on $(\mathrm{Membership})$ and (ΠE) (Table 4), and *Neural Wani* achieves a $1.41\times$ average speedup in the regime where the baseline tends to time out (Table 5).

6 Related Work

Logic-based NLI and automated reasoning. Combining compositional semantics with automated reasoning to perform NLI has a long tradition. Blackburn and Bos (2005) established the canonical pipeline of computational semantics, in which surface strings are compositionally mapped to first-order logic via λ-calculus, and semantic judgments such as entailment and consistency are decided by running off-the-shelf first-order theorem provers and model builders in parallel; *wani*'s YES/NO/UNK procedure (Figure 5) follows the same entailment/contradiction/unknown decision structure. This model-theoretic line was scaled to real-world text through wide-coverage CCG-based semantic construction (Bos et al., 2004). A complementary, proof-theoretic tradition instead establishes entailment by symbolic inference rather than by checking models: natural-logic-based systems (MacCartney and Manning, 2008), tableau-based systems (Abzianidze, 2015), and Coq-based systems (Chatzikyriakidis and Luo, 2014; Mineshima et al., 2015; Martínez-Gómez et al., 2017; Chatzikyriakidis and Bernardy, 2019). Our work belongs to this latter, proof-theoretic branch: in contrast to the model-theoretic tradition that verifies inferences against external models, DTS computes entailment purely through proof search (Section 1.2), and we accelerate the dedicated DTT prover *wani* by guiding its proof search with a neural model rather than leaving it to search purely symbolically.

Neural guidance for theorem proving. Prior attempts to bring learning into theorem proving are largely distinguished by which component of the prover the learned model informs. A first and well-studied locus is premise selection, where a model scores how likely each available fact is to contribute to a proof: Wang et al. (2017) learn deep graph embeddings of formulas for this purpose, and Kogkalidis et al. (2024) pursue the same goal for dependently typed Agda programs via structure-aware encoders. A second locus is the inference engine itself, which can be made differentiable so that symbol-level similarities are learned end to end, as in the differentiable prover of Rocktäschel and Riedel (2017) for knowledge-base completion. *Neural Wani* departs from both: the premises of an NLI problem are already supplied explicitly by the upstream semantic-composition stage, so premise selection is not the bottleneck, and *Neural Wani* leaves *wani*'s symbolic inference engine unchanged rather than replacing it with a differentiable approximation. Instead, our learned model operates at a third locus—the order in which inference rules are expanded during backward search—thereby steering the search while leaving the symbolic core of the prover, and hence its soundness, intact.

7 Conclusion

In this study, we developed *Neural Wani*, which optimizes the proof search process of the DTT-based theorem prover *wani* through neural rule prediction. By focusing on judgments lacking proof terms, we addressed a critical bottleneck in practi-

cal automated theorem proving. Our experimental results confirmed that prioritizing inference rules based on predicted probabilities significantly reduces search time in a natural language inference pipeline, achieving a maximum average speedup of 1.41×.

Limitations

A primary limitation of this study is the discrepancy between the training data and the actual task environment. Ideally, a model for accelerating proof search should be trained on successful proof search trajectories. However, due to the current scarcity of solved proof search instances in complex DTT-based NLI, we used existing type-checking data as a proxy for training. While this allows the model to learn the structural relationships between judgments and rules, there is an inherent distribution shift between type checking and backward inference.

To address this, several concurrent projects (Tomita and Bekki, 2026; Saeki et al., 2026; Sakuma et al., 2026; Kobayashi et al., 2026; Matsubara et al., 2026) are currently underway to expand the coverage of solvable proof search problems within the *wani* ecosystem. We expect that as the volume of successfully solved cases increases, we will be able to refine the model using authentic proof search data, further narrowing the gap between training and real-world application.

Acknowledgments

This work was supported in part by JST CREST Grant Number JPMJCR2565 and JSPS KAKENHI Grant Number JP23H03452.

References

Lasha Abzianidze. 2015. Towards a wide-coverage tableau method for natural logic. In New Frontiers in Artificial Intelligence: JSAI-isAI 2014 Workshops, LENLS, JURISIN, and GABA, Revised Selected Papers, volume 9067 of Lecture Notes in Artificial Intelligence, pages 66–82. Springer.

Daisuke Bekki. 2014. Representing anaphora with dependent types. In Logical Aspects of Computational Linguistics, volume 8535 of Lecture Notes in Computer Science, pages 14–29. Springer.

Daisuke Bekki and Ai Kawazoe. 2016. Implementing variable vectors in a CCG parser. In Logical Aspects of Computational Linguistics, pages 52–67. Springer.

Daisuke Bekki and Koji Mineshima. 2017. Context-passing and underspecification in dependent type semantics. In Stergios Chatzikyriakidis and Zhaohui Luo, editors, Modern Perspectives in Type-Theoretical Semantics, pages 11–41. Springer.

Patrick Blackburn and Johan Bos. 2005. Representation and Inference for Natural Language: A First Course in Computational Semantics. CSLI Studies in Computational Linguistics. CSLI Publications, Stanford, CA.

Johan Bos, Stephen Clark, Mark Steedman, James R. Curran, and Julia Hockenmaier. 2004. Wide-coverage semantic representations from a CCG parser. In COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics, pages 1240–1246, Geneva, Switzerland. COLING.

Stergios Chatzikyriakidis and Jean-Philippe Bernardy. 2019. A wide-coverage symbolic natural language inference system. In Proceedings of the 22nd Nordic Conference on Computational Linguistics, pages 298–303, Turku, Finland. Linköping University Electronic Press.

Stergios Chatzikyriakidis and Zhaohui Luo. 2014. Natural language inference in Coq. Journal of Logic, Language and Information, 23:441–480.

Hinari Daido and Daisuke Bekki. 2020. Development of an automated theorem prover for the fragment of DTS. In the 17th International Workshop on Logic and Engineering of Natural Language Semantics.

Jonathan Evans and Keith Frankish. 2009. In two minds: Dual processes and beyond. Oxford University Press.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. Neural Computation, 9(8):1735–1780.

Ai Kawazoe, Ribeka Tanaka, Koji Mineshima, and Daisuke Bekki. 2015a. A framework for constructing multilingual inference problem sets: Highlighting similarities and differences in semantic phenomena between English and Japanese. In 1st International Workshop on the Use of Multilingual Language Resources in Knowledge Representation Systems.

Ai Kawazoe, Ribeka Tanaka, Koji Mineshima, and Daisuke Bekki. 2015b. An inference problem set for evaluating semantic theories and semantic processing systems for Japanese. In Proceedings of the Twelfth International Workshop on Logic and Engineering of Natural Language Semantics, pages 67–73.

Honoka Kobayashi, Hinari Daido, and Daisuke Bekki. 2026. Neural DTS: Shizen-gengo-suiron-shisutemu-e-no sōkyoku-bunruiki-no kumikomi (trans. Neural DTS: Incorporating hyperbolic classifiers into natural language inference systems). In Proceedings of the 32nd Annual Conference of the Association for Natural Language Processing.

Konstantinos Kogkalidis, Orestis Melkonian, and Jean-Philippe Bernardy. 2024. Learning structure-aware representations of dependent types. In Advances in Neural Information Processing Systems 37 (NeurIPS 2024).

Bill MacCartney and Christopher D. Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008), pages 521–528, Manchester, UK. Coling 2008 Organizing Committee.

Per Martin-Löf. 1984. Intuitionistic type theory. Bibliopolis.

Pascual Martínez-Gómez, Koji Mineshima, Yusuke Miyao, and Daisuke Bekki. 2017. On-demand injection of lexical knowledge for recognising textual entailment. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, pages 710–720, Valencia, Spain. Association for Computational Linguistics.

Mai Matsubara, Yasunori Hokazono, Mitsuhiro Tsunoda, Koutaro Tamura, Hinari Daido, Asa Tomita, and Daisuke Bekki. 2026. Gengogakuteki-paipurain-ni motozuku datōsei-no kenshō: Kin'yū-tekisuto-o taishō-to-shite (trans. Validity verification based on a linguistic pipeline: A case study of financial texts). In Proceedings of the 32nd Annual Conference of the Association for Natural Language Processing.

Koji Mineshima, Pascual Martínez-Gómez, Yusuke Miyao, and Daisuke Bekki. 2015. Higher-order logical inference with compositional semantics. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2055–2061, Lisbon, Portugal. Association for Computational Linguistics.

Nanako Miyagawa, Sora Tagami, and Daisuke Bekki. 2025. Toward the development of an automated theorem prover "Neural Wani" for dependent type theory (in Japanese). In Proceedings of the 39th Annual Conference of the Japanese Society for Artificial Intelligence, 3G5-GS-6-04.

Tim Rocktäschel and Sebastian Riedel. 2017. End-to-end differentiable proving. In Advances in Neural Information Processing Systems 30 (NIPS 2017).

Koharu Saeki, Asa Tomita, Mai Matsubara, and Daisuke Bekki. 2026. Yesod-ni-yoru nihongo-suiron-shisutemu lightblue-no kaihatsu-kankyō-no kōchiku (trans. Building a development environment for the Japanese inference system lightblue using Yesod). In Proceedings of the 32nd Annual Conference of the Association for Natural Language Processing.

Kana Sakuma, Asa Tomita, and Daisuke Bekki. 2026. Izongata-imiron-ni-yoru nihongo-fukugō-dōshi-no imi-gōsei-to suiron (trans. Semantic composition and inference of Japanese compound verbs based on dependent type semantics). In Proceedings of the 32nd Annual Conference of the Association for Natural Language Processing.

Mark Steedman. 1996. Surface Structure and Interpretation. The MIT Press.

Asa Tomita and Daisuke Bekki. 2026. LLM-o mochiita chishiki-hokan-no tōgō-ni-yoru shizen-gengo-suiron-shisutemu lightblue-no kakuchō (trans. Extending the natural language inference system lightblue by integrating LLM-based knowledge completion). In Proceedings of the 32nd Annual Conference of the Association for Natural Language Processing.

Asa Tomita, Mai Matsubara, Hinari Daido, and Daisuke Bekki. 2025. Natural language inference with CCG parser and automated theorem prover for DTS. In Proceedings of the Second Workshop on the Bridges and Gaps between Formal and Computational Linguistics (BriGap-2), pages 1–7, Düsseldorf, Germany. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in Neural Information Processing Systems.

Mingzhe Wang, Yihe Tang, Jian Wang, and Jia Deng. 2017. Premise selection for theorem proving by deep graph embedding. In Advances in Neural Information Processing Systems 30 (NIPS 2017).

A An Example of a Solved JSeM Problem

One of the eight problems solved by both *wani* and *Neural Wani* in our evaluation is JSeM problem 63 (GeneralizedQuantifier section):

- **Premise:** "Fukusuu no shain ga idou o kibou shite iru" ("Several employees want a transfer.")
- **Hypothesis:** "Idou o kibou shite iru shain ga iru" ("There exists an employee who wants a transfer.")
- **Answer:** yes (entailment).

For this problem, both systems construct identical proof trees of 33 rule applications (3 (ΣI), 10 (ΣE), and 20 (Membership)).