Generalized Tree Edit Distance (GTED): A Faithful Evaluation Metric for Statement Autoformalization

Yuntian Liu [1 *] **Tao Zhu** [1 *] **Xiaoyang Liu** [1 *] **Yu Chen** [1] **Zhaoxuan Liu** [1] **Qingfeng Guo** [1] **Jiashuo Zhang** [1] **Kangjie Bao** [1] **Tao Luo** [1 2 3]

Abstract

Statement autoformalization, the automated translation of statement from natural language into formal languages, has become a subject of extensive research, yet the development of robust automated evaluation metrics remains limited. Existing evaluation methods often lack semantic understanding, face challenges with high computational costs, and are constrained by the current progress of automated theorem proving. To address these issues, we propose **GTED** (*Generalized Tree Edit Distance*), a novel evaluation framework that first standardizes formal statements and converts them into operator trees, then determines the semantic similarity using the eponymous GTED metric. On the miniF2F and ProofNet benchmarks, GTED outperforms all baseline metrics by achieving the highest accuracy and Kappa scores, thus providing the community with a more faithful metric for automated evaluation. The code and experimental results are available at https://github.com/XiaoyangLiu-sjtu/GTED.

1. Introduction

Formal languages like Isabelle (Paulson, 1994), HOL Light (Harrison, 1996), Coq (Barras et al., 1999), and Lean (De Moura et al., 2015; Moura & Ullrich, 2021) have recently garnered significant attention within the mathematical community due to their utility in rigorously verifying the correctness of proofs. Nevertheless, the endeavor of formalizing mathematical content demands substantial time and effort, alongside a profound familiarity with these specialized languages, rendering the process inherently labor-intensive. Therefore, the autoformalization task (Szegedy, 2020), defined as translating theorem statements and proofs from natural language into their formal language counterparts, has become a subject of extensive research.

However, despite rapid advancements in autoformalization, the development of robust automated evaluation metrics remains notably limited. Existing evaluation metrics each exhibits specific limitations. For instance, syntax-based methods, such as Typecheck (Jiang et al., 2023), validate compliance with formal grammar but often overlook semantic content. Text similarity metrics, like BLEU (Papineni et al., 2002), are overly sensitive to lexical variation, failing to capture deeper structural or logical equivalence. While proof-based approaches (Liu et al., 2025a) offer logical rigor, they frequently miss valid formalization due to the limited capabilities of the underlying prover. Concurrently, LLM-assisted evaluations (Ying et al., 2024) contend with reproducibility and high cost concerns. These limitations collectively highlight a significant gap: the absence of a consistent, flexible, and efficient semantic evaluation metric.

In this work, we propose GTED (*Generalized Tree Edit Distance*), a novel evaluation framework based on tree edit distance (Zhang & Shasha, 1989). Our framework follows a three-stage process. First, it standardizes formal statements by leveraging the Lean Language Server. Next, it constructs operator trees (OPTs) by parsing the syntactic structure of each statement. Finally, it enhances the tree edit distance calculation to compute a similarity score. Rather than outputting a binary decision, our framework yields a continuous similarity score in the range [0,1], enabling a more nuanced and interpretable evaluation of semantic similarity.

We demonstrate the superiority of the proposed GTED metric by evaluating it against a suite of baseline metrics on miniF2F and ProofNet. GTED achieves the highest overall performance, securing the best Kappa score on both datasets (0.438 on miniF2F and 0.402 on ProofNet) and reaching the highest accuracy (70.73% on miniF2F and 69.89% on ProofNet). Crucially, unlike metrics that are either too strict (e.g., Identity Match, with low recall) or too lenient (e.g., Typecheck, with low precision), GTED also provides the best balance between precision and recall, establishing a

[*]Equal contribution [1]School of Mathematical Sciences, Shanghai Jiao Tong University [2]Institute of Natural Sciences, MOE-LSC, Shanghai Jiao Tong University [3]CMA-Shanghai, Shanghai Artificial Intelligence Laboratory. Correspondence to: Tao Luo <luotao41@sjtu.edu.cn>.

The second AI for MATH Workshop at the 42nd International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

more reliable and robust measure for semantic alignment.

Our main contributions are as follows:

1. We propose GTED (*Generalized Tree Edit Distance*), a novel evaluation metric that introduces and generalizes the standard tree edit distance for the semantic evaluation of statement autoformalization.
2. We implement a procedure to convert formal statements into operator tree representations by leveraging the Lean Language Server.
3. We demonstrate through extensive experiments that GTED achieves both the highest accuracy and the strongest alignment with human expert judgment on the miniF2F and ProofNet benchmarks.

2. Related Work

Autoformalization. Research in autoformalization, particularly for theorem statements, currently shows diverse approaches. While early methods (Wang et al., 2018; Cunningham et al., 2023) often employ neural machine translation, the transformative progress of Large Language Models (LLMs) now leads to the emergence of three dominant strategies for LLM-based autoformalization. These include exploring the efficacy of few-shot prompting (Wu et al., 2022; Agrawal et al., 2022; Zhou et al., 2024); improving capabilities by fine-tuning LLMs (Gao et al., 2024; Lu et al., 2024b; Liu et al., 2025b) on relevant parallel statements; and integrating retrieval-augmented generation (Zhang et al., 2024) to achieve enhanced performance.

Automated Evaluation. In the realm of automated evaluation, early efforts predominantly employ metrics based on grammatical validity (Jiang et al., 2023) and string similarity (Azerbayev et al., 2023), yet these struggle with semantic understanding. FormalAlign (Lu et al., 2024a) innovatively integrates autoformalization with evaluation by simultaneously generating a formal statement and its corresponding evaluation score. While this represents a significant contribution to the fields of autoformalization and automated evaluation, its scoring mechanism cannot be used as a standalone evaluation metric. Simultaneously, cross-provability (Murphy et al., 2024; Li et al., 2024; Liu et al., 2025a) between formal statements emerges as a widely standard for automated evaluation, but its effectiveness is constrained by the current progress in automated theorem proving.

Operator Trees. Operator trees (OPTs) are a foundational data structure in mathematical information retrieval (MIR), a field dedicated to the effective retrieval of mathematical formulae from digital corpora (Zhong et al., 2022b). The prevalence of OPTs stems from their unique capacity to encode both the two-dimensional symbol layout and the underlying syntactic hierarchy of mathematical notation, which enables robust recognition and semantic understanding (Zanibbi et al., 2002; Zanibbi & Blostein, 2012). By preserving vital information such as operator precedence and operand relationships, OPTs enable a more nuanced understanding of mathematical expressions. This semantic depth is foundational to leading MIR systems like WikiMirs (Gao et al., 2016) and Approach0 (Zhong et al., 2022a), underpinning their ability to perform accurate structural matching and retrieval.

3. Methodology

The methodology follows a three-stage pipeline to compute the similarity between formal statements: Section 3.1 details the syntax standardization process, Section 3.2 describes the construction of operator trees (OPTs) from the normalized statements, and Section 3.3 concludes with the formal mathematical definition and formulation of the GTED metric.

3.1. Syntax Standardization

To overcome the analytical challenges posed by syntactic variations in formal statements, such as optional type declarations and compact multivariable declarations, and enable precise structural decomposition, we first standardize the formal statements prior to constructing the operator trees. This standardization process employs two key transformations: (1) theorem rewriting and (2) variable expansion.

Theorem Rewriting. For any given formal statement, the Lean Language Server inherently performs comprehensive syntax normalization as part of its compilation process. This crucial feature handles tasks like type annotation completion, notation expansion, and general syntax standardization. By programmatically interacting with the language server, we effectively rewrite the original formal statement into a processed, normalized form. This resulting representation retains all necessary semantic information while eliminating superficial syntactic variations, thereby preparing it for subsequent operator tree construction. For example, Figure 1(b) illustrates the rewriting of the original formal statement, specifically by adding the crucial type information $\mathbb{R}$ for x.

Variable Expansion. To standardize variable declaration conventions, we expand compact variable declarations (which allow multiple variables to share a single type annotation) into distinct, individual declarations. This process ensures structural clarity in the resulting operator trees while strictly preserving the original mathematical semantics. For instance, Figure 1, part (b) illustrates this by transforming a compact declaration like ($f\ g : \mathbb{R} \to \mathbb{R}$) into individual declarations ($f : \mathbb{R} \to \mathbb{R}$) and ($g : \mathbb{R} \to \mathbb{R}$).

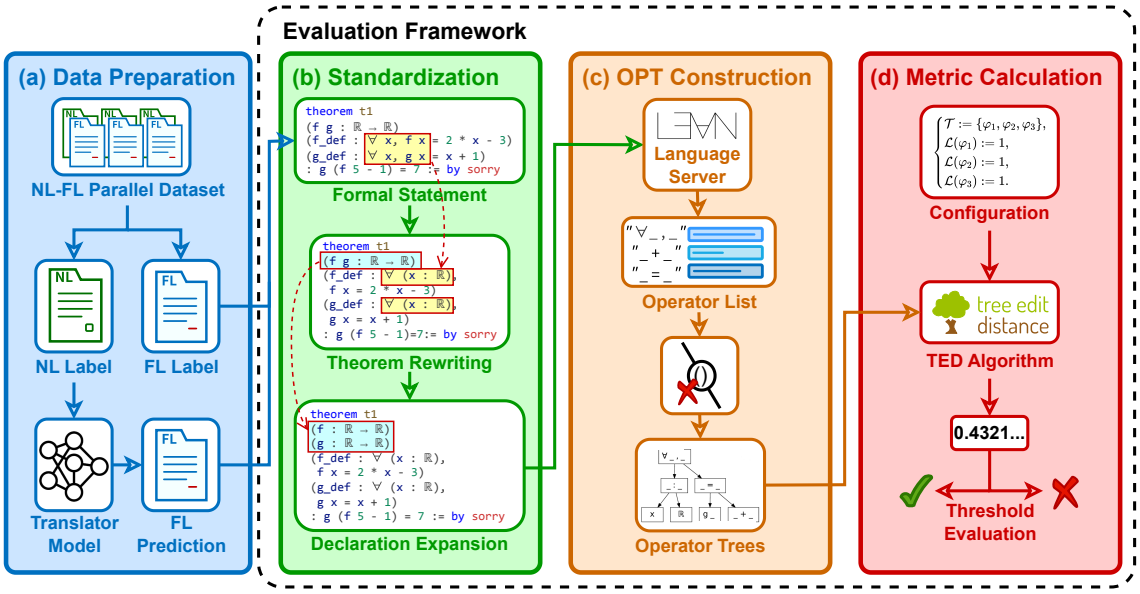

Figure 1. Illustration of GTED (*Generalized Tree Edit Distance*) for statement formalization. (a) Data Preparation: A translator model generates a formal language (FL) prediction from a natural language (NL) input. (b) Standardization: Both the FL prediction and the ground-truth FL label are standardized via theorem rewriting and variable expansion. (c) OPT Construction: The standardized statements are then parsed into operator trees (OPTs) using the Lean Language Server. (d) Metric Calculation: Finally, the GTED is computed between the two OPTs to yield the similarity score.

3.2. OPT Construction

Following standardization, we programmatically interact with the Lean Language Server again to obtain the scope of each element within the formal statement. This scope information is then used to construct an operator tree, integrating two additional operations: (1) placeholder representation and (2) parentheses removal. Figure 2 visually demonstrates a constructed operator tree.

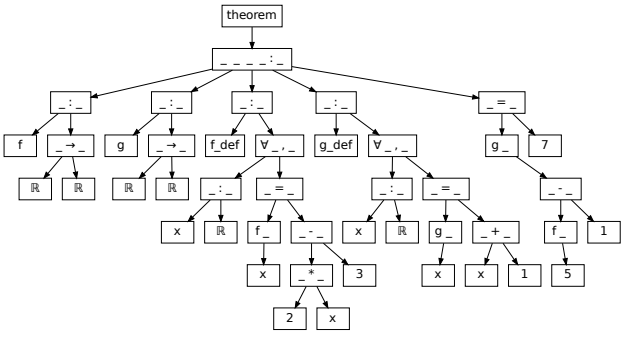

Figure 2. Example of an operator tree for a formal statement.

Placeholder Representation. For the precise capture of structure during operator tree construction and the efficient reconstruction of the original formal statement, we utilize the placeholder representation. This method outlines the tree's components as follows:

- Non-leaf nodes are assigned their operator and associated parameter slots.
- Parameter slots are indicated by underscores (_).
- Leaf nodes represent mathematical objects.

From a long-term perspective, the strategic design of placeholders is crucial because it facilitates the rapid restoration of partial subtrees to their original formal sub-statements. This capability, in turn, enables mutual proof to establish the equivalence of these subtrees. Consequently, this approach seamlessly integrates proof-based evaluation directly into the tree structure. This presents a critical advantage: proving these sub-statements is significantly less challenging than proving the entire statements, thus carving out a key direction for our future work.

Parentheses Removal. Another crucial aspect of OPT construction involves the deliberate elimination of round bracket operators. This approach is driven by a key insight: these brackets are primarily an artifact of linear, string-based representations. In such linear forms, explicit punctuation is necessary to rigorously define grouping and operator precedence when constructing new entities from existing ones.

Conversely, languages inherently represented as trees naturally and explicitly encode these structural dependencies through their topology. Thus, all superfluous round bracket operators are removed from the tree representation, resulting in a more abstract and structurally precise representation that reduces unnecessary computational overhead for subsequent tree edit distance calculations.

3.3. Metric Calculation

Definitions and Notation. We begin by introducing the formal definitions and mathematical notation that provide the foundation for our proposed method.

Definition 1 (Basic Notation)**.** An **Operator Tree** is a rooted tree, where:

- Every leaf node is labeled to represent an independent variable or constant in Lean.
- Every internal node is labeled to represent an operator in Lean, including (dependent) functions and other objects that take multiple arguments.

**Throughout this work, we consider only finite trees.*

Denote that t is a **subtree** of T as $t \subseteq T$.

Define the **Quotient Tree** of an operator tree T and its subtree t, denoted as T/t, as the operator tree that treats t in T as a leaf node.

Definition 2 (Tree Transformations)**.** A **Special Tree Transformation** is defined as a pair of operator trees $f = (T_1, T_2)$, and we denote $f(T_1) = T_2$. Denote the collection of special tree transformations as $\mathcal{S}$ ($\mathcal{S} = \mathcal{T} \times \mathcal{T}$). We can also naturally define the composition of special tree transformations:

$$(T_1, T_2) \circ (T_2, T_3) = (T_1, T_3).$$

Given two special tree transformations $f, g \in \mathcal{S}$ where $f = (t_1, t_2), g = (T_1, T_2)$. Define f is a **Local Depiction** of g, denoted as $f \preceq g$, if and only if:

(1) t_1 and t_2 are respectively subtrees of T_1 and T_2:

$$t_1 \subseteq T_1 \wedge t_2 \subseteq T_2.$$

(2) g can be depicted by f:

$$T_1/t_1 = T_2/t_2.$$

Define f is a **Co-local Depiction** of g, denoted as $f \preccurlyeq g$, if and only if there exists a disjoint list of common subtrees $\tau_1, \tau_2, \ldots, \tau_m$ of T_1 and T_2, such that:

$$\begin{cases} T_1/\tau_1, \ldots, \tau_m = t_1 \\ T_2/\tau_1, \ldots, \tau_m = t_2 \end{cases}$$

We may naturally infer that the local depiction relation is a partial order. For every special tree transformation $f \in \mathcal{S}$, define the **Generalized Tree Transformation** of f, denoted as φ_f, as the collection of all special tree transformations that can be locally depicted by f:

$$\varphi_f := \{g | f \preceq g\} \cup \{g | f \preccurlyeq g\}$$

Denote the collection of all generalized tree transformations as $\mathcal{G}$:

$$\mathcal{G} := \{\varphi_f | f \in \mathcal{S}\}$$

Formulation. Let $\mathcal{H}$ be a disjoint subset of $\mathcal{G}$ that represents the set of allowed transformations, where every two general tree transformations do not intersect. Let $\mathcal{L} : \mathcal{H} \to \mathbb{N}$ be the cost function that assigns a specific cost value in $\mathbb{N}$ (or potentially other number systems depending on the demand) to every general tree transformation available, with $\mathrm{Id} \mapsto 0$. The generalized tree edit distance d_{GTED} between two operator trees T_1 and T_2, with available transformations in Φ and a cost function $\mathcal{L}$, is defined to be the infimum of total cost of all possible finite sequences of special tree transformations that convert T_1 to T_2:

$$d_{\text{GTED}}(\mathcal{H}, \mathcal{L}; T_1, T_2) := \inf \left(\sum_{i=1}^{n} \mathcal{L}(\varphi_{f_i}) \right),$$

where $f_1, \ldots, f_n$ fit $T_2 = f_n \circ \cdots \circ f_1(T_1)$ and $\varphi_{f_i} \in \mathcal{H}$.

For the purpose of binary semantic evaluation, we map the continuous loss value to a discrete outcome. This is achieved by transforming the loss into a normalized similarity score on the interval [0,1] and subsequently applying a decision threshold θ. The first formulation is analogous to the conventional TED similarity, and we refer the interested reader to the comprehensive survey by (Bille, 2005) for more details on the algorithm.

$$\delta_{\text{GTED}}(\theta; T_1, T_2) := \mathbb{I}_{\{\cdot > \theta\}} \left(1 - \frac{d_{\text{GTED}}(\mathcal{H}, \mathcal{L}; T_1, T_2)}{\max(|T_1|, |T_2|)} \right),$$

where $|T|$ stands for the number of vertices in the tree T, and the overall expression serves as an indicator for non-identity, yielding 1 if the trees differ and 0 if they are same.

In practice, $\mathcal{H}$ is likely to include: (1) "legal" transformations, which depict the effect of proof steps in Lean, with certain restrictions in order for better alignment with human intuition; (2) "dumb" transformations, like those in the original TED algorithm to guarantee a probably reasonable result when a formal proof is unattainable. When no "dumb" transformations are allowed, there may be no proper sequences available, in which case the total loss is $+\infty$, which in a binary evaluation process gives a negative result directly. The normalization process may also fail.

Now the standard TED algorithm can be considered as a special case of our GTED metric if we let $\mathcal{G} = \{\text{Inserting}, \text{Deleting}, \text{Relabeling}\}$ and normally set $\mathcal{L} := \varphi \mapsto 1$. So theoretically, $\mathcal{H}$, $\mathcal{L}$, and θ can be deliberately designed according to demand.

α-conversion. To serve as an example for the generalized tree transformations, we choose one of the most reasonable operations in formal mathematics, renaming the names of variables or hypotheses, which corresponds to α-conversion in the language of λ-calculus, one of the foundational theories of Lean. α-conversion says that, renaming the same variable in a formula does not change its essence:

$$\boxed{\lambda(x).f[x] \overset{x\mapsto y}{=\!=\!=} \lambda(y).f[y]}.$$

Besides regular λ-abstractions, numerous common operators like dependent function types ($\Pi_{(x:\alpha)}, \beta(x)$), universal and existential quantifiers ($\forall, \exists$) all behave similarly in Lean, since they're also defined with λ-abstractions. In actual syntax of λ-calculus, the name of a variable is overwritten if undergone another λ-abstraction, as demonstrated in the example below, where the x's are considered different so that the inner x overwrites the outer one, thus giving z as the result instead of y:

$$\boxed{\lambda(x).(\lambda(x).x)yz = z}.$$

In Lean, an α-conversion between two statements can be exemplified as follows (assuming `P` is a predefined predicate with type `Nat -> Prop`):

```
theorem t1 (x : Nat) : P x := by sorry
theorem t2 (y : Nat) : P y := by sorry
```

Under the framework of GTED, α-conversion can be rebuilt as a generalized tree transformation, which allows it to be handled as a formal proof step that rigorously preserves mathematical semantics.

4. Experiments

4.1. Experiment Setting

Dataset. For evaluation, we use the miniF2F-test (Zheng et al., 2022) and ProofNet-test (Azerbayev et al., 2023) datasets. The specific versions employed in this study are sourced from Numina[1] for miniF2F-test, and from DeepSeek[2] for ProofNet-test. Moreover, since automated evaluation requires both ground truth and predicted formal statements, we employ HERALD Translator (Gao et al., 2024), a state-of-the-art autoformalization model, to translate these datasets into their respective formal statements for subsequent evaluation.

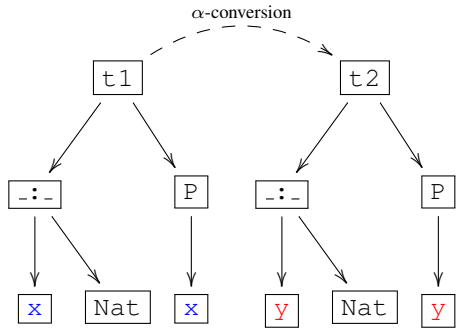

Figure 3. An example of α-conversion, the renaming of a bound variable.

Baseline Metrics. We benchmark GTED against several competing baselines metrics to validate our proposed metric's efficacy. To ensure fair and consistent evaluation across these diverse metrics, we establish specific conventions for handling theorem names. For string-matching metrics (e.g., Identity Match, BLEU), we standardize theorem names in both ground truth and predicted formal statements to `thm`. Conversely, for proof-based metrics (e.g., BEq), we designate the ground truth theorem names as `thm_P` and the predicted theorem names as `thm_Q`. For all other metrics, theorem names remain unaltered.

- Identity Match: This metric considers a predicted formal statement as correct if, after removing all whitespace, it is identical to the ground truth.
- Typecheck: A predicted formal statement is deemed correct if it successfully compiles.
- BLEU: We align with the ProofNet (Azerbayev et al., 2023) for the computation.
- Majority Voting: Following the setup described in Lean Workbook (Ying et al., 2024) and BEq (Liu et al., 2025a), we employ DeepSeek-V3 (Liu et al., 2024) with temperature 0.7 for 16 rounds of majority voting.
- Definitional Equality (Liu et al., 2025a): This metric directly attempts to prove `example: thm_P = thm_Q := by rfl`. If the proof is successful, the predicted formal statement is considered correct.
- BEq (Liu et al., 2025a): This metric involves attempting to prove `thm_Q` using `thm_P` and vice-versa, employing a broader range of tactics. A predicted formal statement is considered correct if both directions of proof are successfully completed.

Regarding tactic usage for BEq, we adopt the "Normal

[1]https://huggingface.co/datasets/AI-MO/minif2f_test

[2]https://github.com/deepseek-ai/DeepSeek-Prover-V1.5/tree/main/datasets

Table 1. Performance of automated evaluation metrics for statement formalization. Detailed results are available in Appendix A.1.

Metric	miniF2F				ProofNet			
	Precision	Recall	Accuracy	Kappa	Precision	Recall	Accuracy	Kappa
Identity Match	**100.00%**	11.48%	47.32%	0.095	0.00%	0.00%	47.31%	0.000
Typecheck	59.51%	**100.00%**	59.51%	0.000	52.69%	**100.00%**	52.69%	0.000
BLEU	78.22%	64.75%	68.29%	0.368	72.34%	69.39%	**69.89%**	0.398
Majority Voting	88.00%	54.10%	68.29%	0.397	77.78%	57.14%	68.82%	0.384
Definitional Equality	**100.00%**	36.07%	61.95%	0.314	60.00%	6.12%	48.39%	0.015
BEq	98.28%	46.72%	67.80%	0.405	**100.00%**	16.33%	55.91%	0.156
GTED	88.75%	58.20%	**70.73%**	**0.438**	75.61%	63.27%	**69.89%**	**0.402**

set" of tactics {`exact`, `exact?`, `have`, `apply`, `cases'`, `constructor`, `ext`, `intro`, `intros`, `rw`, `use`}, as this set demonstrates the best performance in the original paper. Furthermore, we observe that a sampling configuration of temperature T=0.1 and top_p=0.9 leads to more successful bidirectional proofs compared to the beam search used in the original paper (miniF2F: 58 vs 53, ProofNet: 8 vs 8). Consequently, we adopt this sampling setting for our experiments.

Human Evaluation. To establish the ground truth labels, the miniF2F-test and ProofNet-test datasets are first translated using the HERALD Translator. From these translated results, 205 entries from miniF2F-test and 93 from ProofNet-test successfully compile. Four Lean4 human experts then judge the correctness of these predicted formal statements. Subsequently, different metrics are evaluated by comparing their results against these human-expert judgments, reporting Precision, Recall, Accuracy, and Kappa scores.

Implementation Details. A fully-featured implementation of α-conversion requires both variable renaming and scope-awareness to ensure consistency. Due to the time constraints of the present work, we have provisionally deferred the implementation of the scope-aware aspect and implemented only the primary renaming operation.

All experiments in this work are conducted using the Lean toolchain v4.19.0-rc2 on a single NVIDIA A100 GPU with 40GB of memory.

4.2. Experiment Results

Overall Comparision. Table 1 presents a comprehensive overview of our proposed GTED metric compared to various baseline metrics across the miniF2F and ProofNet datasets. Our primary focus for evaluation is on **Kappa** and **Accuracy**, as Kappa (Cohen, 1960) offers a robust measure of agreement beyond chance, and Accuracy directly reflects the overall correctness of the evaluation. GTED consistently outperforms all other metrics in these crucial aspects, demonstrating its superior ability to reliably assess formal statements. It achieves the highest Kappa scores (**0.438** on miniF2F and **0.402** on ProofNet), indicating a **moderate level of agreement** (0.4-0.6) with human expert evaluations across both datasets. This consistent performance underscores GTED's robustness, a quality not matched by other metrics, which often fail to maintain such agreement. Furthermore, GTED leads in Accuracy (**70.73%** on miniF2F and **69.89%** on ProofNet).

Comparison with BLEU. Both GTED and BLEU metrics produce continuous-valued outputs requiring thresholding for binary classification. While Table 1 reports their highest Kappa scores across all thresholds, Figure 4 highlights a key advantage of GTED in terms of its **robustness to threshold selection**. BLEU's optimal Kappa is confined to a narrow threshold window, with slight deviations causing significant performance drops across all metrics. This sensitivity makes BLEU challenging to deploy reliably, as identifying and fixing the precise optimal threshold in practice is difficult. Conversely, GTED exhibits a robust performance plateau, maintaining high and consistent metric values over a much broader threshold range. This "flatness" renders GTED a more practical and dependable evaluation metric, a phenomenon similarly observed on ProofNet (see Appendix A.4), reducing the burden of threshold selection and enhancing result generalizability.

Comparison with Majority Voting. GTED provides a compelling alternative to LLM-based evaluation methods like Majority Voting. As shown in Table 1, our approach is not only competitive but superior in performance, outperforming Majority Voting in both accuracy (e.g., 70.73% vs. 68.29% on miniF2F) and Kappa score (0.438 vs. 0.397). Beyond its higher accuracy, our method operates at a significantly lower computational cost. This combination of being lightweight yet more reliable makes GTED highly advantageous in scenarios where inference efficiency and deployment simplicity are critical.

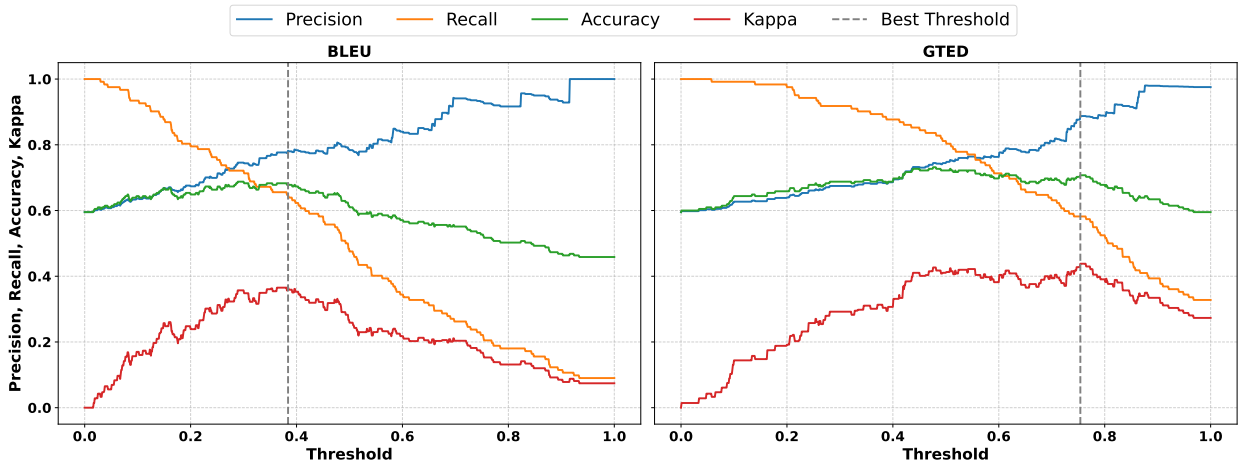

Figure 4. Comparison of BLEU and GTED across thresholds on miniF2F.

Comparison with Proof-based Metrics. Comparing GTED with proof-based metrics like Definitional Equality and BEq, Table 1 reveals key distinctions. While Definitional Equality and BEq show strong Precision, their low Recall and Kappa scores, particularly on the more challenging ProofNet dataset, indicate an overly conservative approach that misses many correct formalization. Moreover, the current development of automated theorem proving (ATP) exacerbates this issue. For instance, the SOTA proving model, DeepSeek-Prover-V2-671B (Ren et al., 2025), currently achieves only a 7.45% proof rate on the PutnamBench dataset (Tsoukalas et al., 2024). This low proof rate implies that many essentially equivalent formalization may not be provable by automated means, thereby leading to a high rate of false negatives. In contrast, GTED offers a more balanced, comprehensive, and robust assessment without being constrained by ATP's current development.

Additionally, we unexpectedly identify **three instances of false positives within the proof-based metrics**, which are further elaborated upon in Appendices A.2 and A.3.

5. Conclusion and Future Work

We propose GTED (*Generalized Tree Edit Distance*), a novel framework that assesses semantic similarity via the edit distance between operator trees of formal statements. Experiments on the miniF2F and ProofNet benchmarks show that GTED consistently outperforms all baseline metrics, achieving the highest accuracy and Kappa scores, which demonstrates a superior alignment with human expert judgment. We conclude that GTED provides the community with a more faithful metric for automated evaluation.

Our future work will focus on enhancing GTED by incorporating deeper mathematical domain knowledge. The immediate priority is to complete our implementation of α-conversion by making it fully scope-aware, which will improve the handling of bound variables. More broadly, we aim to address the primary limitation of the current metric: its "semantic naivety," which incorrectly penalizes logically equivalent yet syntactically different expressions (e.g., $x + y$ and $y + x$). To overcome this, we will develop a semantically-aware framework by augmenting GTED with a system of zero- or low-cost rewrite rules derived from fundamental mathematical axioms and theorems. These enhancements will ultimately transform GTED into a more robust and adaptable evaluation tool, capable of gauging true mathematical meaning beyond superficial syntactic structure.

References

Agrawal, A., Gadgil, S., Goyal, N., Narayanan, A., and Tadipatri, A. Towards a mathematics formalisation assistant using large language models. *arXiv preprint arXiv:2211.07524*, 2022.

Azerbayev, Z., Piotrowski, B., Schoelkopf, H., Ayers, E. W., Radev, D., and Avigad, J. ProofNet: Autoformalizing and formally proving undergraduate-level mathematics. *arXiv preprint arXiv:2302.12433*, 2023.

Barras, B., Boutin, S., Cornes, C., Courant, J., Coscoy, Y., Delahaye, D., de Rauglaudre, D., Filliâtre, J.-C., Giménez, E., Herbelin, H., et al. The Coq proof assistant reference manual. *INRIA, version*, 6(11), 1999.

Bille, P. A survey on tree edit distance and related problems. *Theoretical Computer Science*, 337(1-3):217–239, 2005.

Cohen, J. A coefficient of agreement for nominal scales.

Educational and Psychological Measurement, 20(1):37–46, 1960.

Cunningham, G., Bunescu, R. C., and Juedes, D. Towards autoformalization of mathematics and code correctness: Experiments with elementary proofs. In *Proceedings of the 1st Workshop on Mathematical Natural Language Processing (MathNLP)*, pp. 25–32. Association for Computational Linguistics, 2023.

De Moura, L., Kong, S., Avigad, J., Van Doorn, F., and von Raumer, J. The Lean theorem prover (system description). In *Automated Deduction-CADE-25: 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings 25*, pp. 378–388. Springer, 2015.

Gao, G., Wang, Y., Jiang, J., Gao, Q., Qin, Z., Xu, T., and Dong, B. Herald: A natural language annotated Lean 4 dataset. In *The Thirteenth International Conference on Learning Representations*, 2024.

Gao, L., Yuan, K., Wang, Y., Jiang, Z., and Tang, Z. The Math retrieval system of ICST for NTCIR-12 MathIR task. In *NTCIR*, 2016.

Harrison, J. HOL Light: A tutorial introduction. In *International Conference on Formal Methods in Computer-Aided Design*, pp. 265–269. Springer, 1996.

Jiang, A. Q., Li, W., and Jamnik, M. Multilingual mathematical autoformalization. *arXiv preprint arXiv:2311.03755*, 2023.

Li, Z., Wu, Y., Li, Z., Wei, X., Zhang, X., Yang, F., and Ma, X. Autoformalize mathematical statements by symbolic equivalence and semantic consistency. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Liu, A., Feng, B., Xue, B., Wang, B., Wu, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Liu, Q., Zheng, X., Lu, X., Cao, Q., and Yan, J. Rethinking and improving autoformalization: Towards a faithful metric and a dependency retrieval-based approach. In *The Thirteenth International Conference on Learning Representations*, 2025a.

Liu, X., Bao, K., Zhang, J., Liu, Y., Chen, Y., Liu, Y., Jiao, Y., and Luo, T. ATLAS: Autoformalizing Theorems through Lifting, Augmentation, and Synthesis of Data. *arXiv preprint arXiv:2502.05567*, 2025b.

Lu, J., Wan, Y., Huang, Y., Xiong, J., Liu, Z., and Guo, Z. Formalalign: Automated alignment evaluation for autoformalization. In *The Thirteenth International Conference on Learning Representations*, 2024a.

Lu, J., Wan, Y., Liu, Z., Huang, Y., Xiong, J., Liu, C., Shen, J., Jin, H., Zhang, J., Wang, H., et al. Process-driven autoformalization in Lean 4. *arXiv preprint arXiv:2406.01940*, 2024b.

Moura, L. d. and Ullrich, S. The Lean 4 theorem prover and programming language. In *Automated Deduction–CADE 28: 28th International Conference on Automated Deduction, Virtual Event, July 12–15, 2021, Proceedings 28*, pp. 625–635. Springer, 2021.

Murphy, L., Yang, K., Sun, J., Li, Z., Anandkumar, A., and Si, X. Autoformalizing euclidean geometry. In *Forty-first International Conference on Machine Learning*, 2024.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pp. 311–318, 2002.

Paulson, L. C. *Isabelle: A Generic Theorem Prover*. Springer, 1994.

Ren, Z., Shao, Z., Song, J., Xin, H., Wang, H., Zhao, W., Zhang, L., Fu, Z., Zhu, Q., Yang, D., et al. Deepseek-prover-v2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *arXiv preprint arXiv:2504.21801*, 2025.

Szegedy, C. A promising path towards autoformalization and general artificial intelligence. In *Intelligent Computer Mathematics: 13th International Conference, CICM 2020, Bertinoro, Italy, July 26–31, 2020, Proceedings 13*, pp. 3–20. Springer, 2020.

Tsoukalas, G., Lee, J., Jennings, J., Xin, J., Ding, M., Jennings, M., Thakur, A., and Chaudhuri, S. PutnamBench: Evaluating neural theorem-provers on the Putnam mathematical competition. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

Wang, Q., Kaliszyk, C., and Urban, J. First experiments with neural translation of informal to formal mathematics. In *Intelligent Computer Mathematics: 11th International Conference, CICM 2018, Hagenberg, Austria, August 13-17, 2018, Proceedings 11*, pp. 255–270. Springer, 2018.

Wu, Y., Jiang, A. Q., Li, W., Rabe, M., Staats, C., Jamnik, M., and Szegedy, C. Autoformalization with large language models. *Advances in Neural Information Processing Systems*, 35:32353–32368, 2022.

Ying, H., Wu, Z., Geng, Y., Wang, J., Lin, D., and Chen, K. Lean Workbook: A large-scale Lean problem set formalized from natural language math problems. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

Zanibbi, R. and Blostein, D. Recognition and retrieval of mathematical expressions. *International Journal on Document Analysis and Recognition (IJDAR)*, 15:331–357, 2012.

Zanibbi, R., Blostein, D., and Cordy, J. R. Recognizing mathematical expressions using tree transformation. *IEEE Transactions on pattern analysis and machine intelligence*, 24(11):1455–1467, 2002.

Zhang, K. and Shasha, D. Simple fast algorithms for the editing distance between trees and related problems. *SIAM journal on computing*, 18(6):1245–1262, 1989.

Zhang, L., Quan, X., and Freitas, A. Consistent autoformalization for constructing mathematical libraries. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 4020–4033. Association for Computational Linguistics, November 2024.

Zheng, K., Han, J. M., and Polu, S. miniF2F: a cross-system benchmark for formal Olympiad-level mathematics. In *International Conference on Learning Representations*, 2022.

Zhong, W., Xie, Y., and Lin, J. Applying structural and dense semantic matching for the arqmath lab 2022, clef. In *CLEF (Working Notes)*, pp. 147–170, 2022a.

Zhong, W., Yang, J.-H., Xie, Y., and Lin, J. Evaluating token-level and passage-level dense retrieval models for math information retrieval. *arXiv preprint arXiv:2203.11163*, 2022b.

Zhou, J. P., Staats, C. E., Li, W., Szegedy, C., Weinberger, K. Q., and Wu, Y. Don't Trust: Verify – grounding LLM quantitative reasoning with autoformalization. In *The Twelfth International Conference on Learning Representations*, 2024.

A. Additional Experimental Analysis

A.1. Detailed Experimental Results

Tables 2 and 3 present the detailed experimental results, where TP, TN, FP, and FN denote the number of true positives, true negatives, false positives, and false negatives, respectively. Notably, BEq exhibits one false positives in the miniF2F dataset, while Definitional Equality also records two false positives in ProofNet.

Table 2. Detailed experimental results of automated evaluation metrics on miniF2F.

Metric	miniF2F							
	TP	TN	FP	FN	Precision	Recall	Accuracy	Kappa
Identity Match	14	83	0	108	**100.00%**	11.48%	47.32%	0.095
Typecheck	122	0	83	0	59.51%	**100.00%**	59.51%	0.000
BLEU	79	61	22	43	78.22%	64.75%	68.29%	0.368
Majority Voting	66	74	9	56	88.00%	54.10%	68.29%	0.397
Definitional Equality	44	83	0	78	**100.00%**	36.07%	61.95%	0.314
BEq	57	82	1	65	98.28%	46.72%	67.80%	0.405
GTED	71	74	9	51	88.75%	58.20%	**70.73%**	**0.438**

Table 3. Detailed experimental results of automated evaluation metrics on ProofNet.

Metric	ProofNet							
	TP	TN	FP	FN	Precision	Recall	Accuracy	Kappa
Identity Match	0	44	0	49	0/0	0.00%	47.31%	0.000
Typecheck	49	0	44	0	52.69%	**100.00%**	52.69%	0.000
BLEU	34	31	13	15	72.34%	69.39%	**69.89%**	0.398
Majority Voting	28	36	8	21	77.78%	57.14%	68.82%	0.384
Definitional Equality	3	42	2	46	60.00%	6.12%	48.39%	0.015
BEq	8	44	0	41	**100.00%**	16.33%	55.91%	0.156
GTED	31	34	10	18	75.61%	63.27%	**69.89%**	**0.402**

A.2. Two FP Cases for Definitional Equality

 # exercise_1_3_8
NL: Prove that if $\Omega = \{1, 2, 3, \ldots\}$ then S_Ω is an infinite group.

Label:
```
theorem thm_P : Infinite (Equiv.Perm ℕ) := by sorry
```

Prediction:
```
theorem thm_Q : {Ω : Type u_1} [Infinite Ω] : Infinite (Equiv.Perm Ω) := by sorry
```

Definitional Equality
```
example : thm_P = thm_Q := by rfl
```

In Prediction, the original proposition is generalized, resulting in a semantic difference. However, in Definition Equality, the Lean compiler will automatically infer that Ω in Prediction is the set of natural numbers $\mathbb{N}$, thus passing the verification and leading to a false positive.

 # exercise_9_4_2c
NL: Prove that $x^4 + 4x^3 + 6x^2 + 2x + 1$ is irreducible in $\mathbb{Z}[x]$.

Label:
```
theorem thm_P : Irreducible  (X^4 + 4*X^3 + 6*X^2 + 2*X + 1 : Polynomial ℤ) := by
    sorry
```

Prediction:
```
theorem thm_Q : Irreducible (wilsons_poly : ℤ[X]) := by sorry
```

Definitional Equality
```
example : thm_P = thm_Q := by rfl
```

In the Prediction, an undefined variable wilsons_poly appeared instead of the polynomial given in the problem, resulting in a semantic difference. However, Lean will automatically interpret wilsons_poly as an implicit variable. Therefore, in Definitional Equality, the Lean compiler will automatically infer wilsons_poly as the polynomial $X^4+4X^3+6X^2+2X+1$, thus passing the verification and leading to a false positive.

A.3. One FP Case for BEq

 # mathd_numbertheory_254
NL: Sally, Wei-Hwa, and Zoe are playing a game of marbles involving first arranging as many piles of 10 marbles as possible. Sally brought 239 marbles, Wei-Hwa brought 174 marbles, and Zoe brought 83 marbles. If all their marbles are grouped together, how many must be removed in order to start the game? Show that it is 6.

Label:
```
theorem thm_P : (239 + 174 + 83) % 10 = 6 := by sorry
```

Prediction:
```
theorem thm_Q (s w z : ℕ) : (239 + 174 + 83) % 10 = 6 := by sorry
```

BEq: Label → Prediction
```
theorem thm_Q (s w z : ℕ) : (239 + 174 + 83) % 10 = 6 := by
  exact thm_P
```

BEq: Prediction → Label
```
theorem thm_P : (239 + 174 + 83) % 10 = 6 := by
  exact thm_Q 239 174 83
```

The prediction is considered incorrect as it contains redundant variables `(s w z : ℕ)`. Human evaluators penalize such outputs, because the introduction of superfluous variables can cause confusion and represents an undesirable behavior for translator models.

A.4. Performance of BLEU and GTED on ProofNet

Figure 5 shows the performance of BLEU and GTED across thresholds on ProofNet.

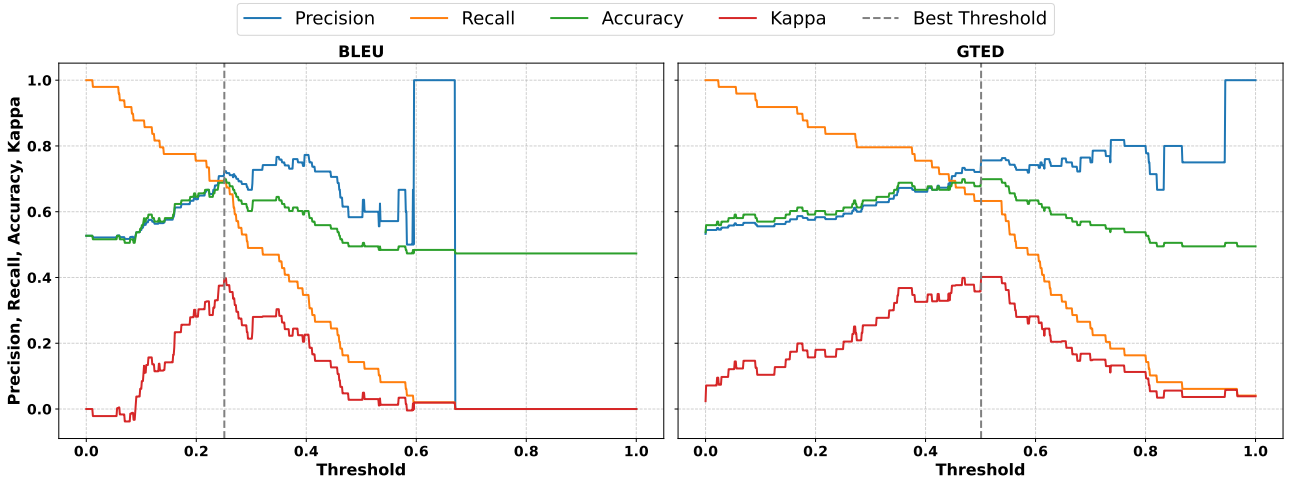

Figure 5. Comparison of BLEU and GTED across thresholds on ProofNet.