ASSESS: A Semantic and Structural Evaluation Framework for Statement Similarity

Anonymous authors
Paper under double-blind review

Abstract

Statement autoformalization, the automated translation of statements from natural language into formal languages, has seen significant advancements, yet the development of automated evaluation metrics remains limited. Existing metrics for formal statement similarity often fail to balance semantic and structural information. String-based approaches capture syntactic structure but ignore semantic meaning, whereas proof-based methods validate semantic equivalence but disregard structural nuances and, critically, provide no graded similarity score in the event of proof failure. To address these issues, we introduce **ASSESS** (*A Semantic and Structural Evaluation Framework for Statement Similarity*), which comprehensively integrates semantic and structural information to provide a continuous similarity score. Our framework first transforms formal statements into Operator Trees to capture their syntactic structure and then computes a similarity score using our novel **TransTED** (*Transformation Tree Edit Distance*) **Similarity** metric, which enhances traditional Tree Edit Distance by incorporating semantic awareness through transformations. For rigorous validation, we present **EPLA** (*Evaluating Provability and Likeness for Autoformalization*), a new benchmark of 524 expert-annotated formal statement pairs derived from miniF2F and ProofNet, with labels for both semantic provability and structural likeness. Experiments on EPLA demonstrate that TransTED Similarity outperforms existing methods, achieving state-of-the-art accuracy and the highest Kappa coefficient. The benchmark and experimental results are provided in the supplementary material.

1 Introduction

Formal languages such as Isabelle (Paulson, 1994), HOL Light (Harrison, 1996), Coq (Barras et al., 1999), and Lean (De Moura et al., 2015; Moura & Ullrich, 2021) have recently gained prominence within the mathematical community for their capacity to rigorously verify proofs. Nevertheless, formalizing mathematical content is a labor-intensive process that demands substantial time, effort, and a profound familiarity with these specialized languages and their corresponding mathematical libraries, such as Mathlib (The mathlib Community, 2020). Consequently, the task of autoformalization (Szegedy, 2020), defined as translating theorem statements and proofs from natural language into their formal counterparts, has become an active area of research.

While autoformalization has rapidly advanced, the methods for evaluating its output have not kept pace. Existing evaluation metrics can be unified under the framework of assigning a similarity score, and they are constrained by a fundamental trade-off between capturing semantic meaning and preserving structural information. String-based metrics such as BLEU (Papineni et al., 2002) rely on surface-level n-gram overlap. This makes them sensitive to inconsequential lexical variations but largely ignores underlying semantic content. Conversely, proof-based approaches (Li et al., 2024; Liu et al., 2025a) can guarantee semantic equivalence through formal proof. However, this method is brittle; it is fundamentally constrained by the capabilities of the underlying automated theorem prover, leading to a high false negative rate. Furthermore, when a proof cannot be found, these approaches fail to provide any meaningful similarity score. In addition, syntax-based metrics (Jiang et al., 2023) operate at the shallowest level of grammatical compliance, offering no semantic or structural insights. The LLM-as-a-Judge (Ying et al., 2024a) approach, while powerful, introduces prohibitive costs and significant reproducibility concerns. These collective shortcomings reveal a critical gap in the field: the need for an automated evaluation metric that robustly combines semantic and structural information for statement similarity in a reproducible and efficient manner.

In this work, we propose **ASSESS** (*A Semantic and Structural Evaluation Framework for Statement Similarity*), a novel two-stage framework for evaluating formal statement pairs. The core of ASSESS is its novel metric, **TransTED** (*Transformation Tree Edit Distance*) **Similarity**. It is designed to be sensitive to both semantic and structural nuances while remaining computationally efficient, requiring only CPU resources. In the first stage, ASSESS leverages the Lean Language Server to parse each formal statement pair into its Operator Tree, a representation that captures deeper structural information than raw text. To overcome the semantic limitations of a standard Tree Edit Distance (TED) (Zhang & Shasha, 1989), the second stage augments the TED calculation with a curated set of transformations. This augmentation enables TransTED Similarity to robustly measure statement similarity where purely structural or semantic methods fail.

To enable a rigorous evaluation in the absence of a standard benchmark, we introduce **EPLA** (*Evaluating Provability and Likeness for Autoformalization*). We constructed this benchmark by first translating informal statements from the miniF2F-test (Zheng et al., 2022) and ProofNet-test (Azerbayev et al., 2023) datasets using the Herald Translator (Gao et al., 2024) and Gemini 2.5 Pro (Comanici et al., 2025). Successful translations were then paired with their ground-truth counterparts and subsequently annotated by human experts along two axes: semantic provability and structural likeness. The resulting benchmark comprises 524 annotated pairs, drawn from EPLA-miniF2F (373 pairs) and EPLA-ProofNet (151 pairs). To our knowledge, this represents the largest and most comprehensive dataset dedicated to this evaluation task.

Experimental results demonstrate that our proposed TransTED Similarity metric consistently outperforms a range of baselines, establishing a new state-of-the-art in both accuracy (78.82% and 70.86%) and the Cohen's Kappa coefficient (0.46 and 0.40) across both the EPLA-miniF2F and EPLA-ProofNet datasets. Our metric provides a more balanced assessment than brittle proof-based methods, is more efficient and reproducible than LLM-based judges, and exhibits superior robustness compared to n-gram-based metrics. Finally, a detailed ablation study confirms that our novel transformation component is the critical factor behind these performance gains, validating its effectiveness in capturing semantic equivalence. This highlights its success in providing a more balanced and reliable measure of statement similarity.

We summarize our main contributions as follows:

1. We propose a novel method that leverages the Lean Language Server to automatically parse formal statements into Operator Trees. This representation enables the direct application of Tree Edit Distance (TED) similarity for evaluating statement similarity.
2. We introduce TransTED Similarity, a semantically-aware metric that extends TED similarity by incorporating transformations directly into the distance calculation. TransTED similarity provides a more holistic similarity score, establishing a new state-of-the-art.
3. We present EPLA, the first large-scale and most comprehensive benchmark for this evaluation task, comprising 524 formal statement pairs meticulously annotated by domain experts, to facilitate rigorous and reproducible evaluation.

2 Related Work

Autoformalization. The goal of autoformalization is to translate informal mathematics into formally verified code. Research in this area, particularly for theorem statements, has evolved significantly. While early work often relied on neural machine translation (Wang et al., 2018; Cunningham et al., 2023), the transformative impact of Large Language Models (LLMs) has given rise to three dominant strategies. 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; Wu et al., 2025) on relevant parallel statements; and integrating retrieval-augmented generation (Zhang et al., 2024) to achieve enhanced performance.

Automated Evaluation. Automated evaluation in the context of autoformalization denotes methods and metrics that assign quantitative scores to machine-generated formal statements, with the goal of estimating their similarity relative to a natural-language source (Weng et al., 2025). In the

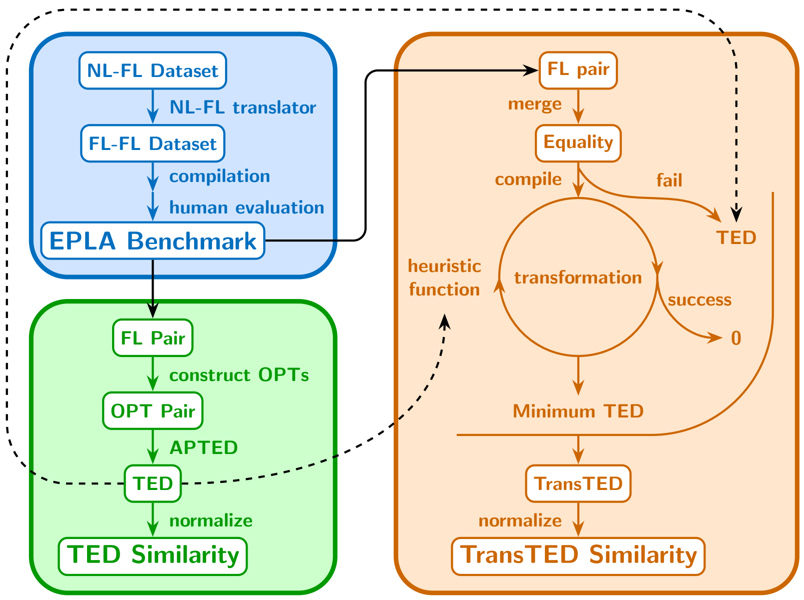

Figure 1: Illustration of ASSESS. (a) EPLA Benchmark: The dataset is created by using translators to generate formal pairs, which are processed through compilation and human evaluation. (b) TED Similarity: The baseline TED Similarity is computed by converting an FL pair into Operator Trees, calculating and normalizing the TED. (c) TransTED Similarity: The novel TransTED Similarity first merges an FL pair into an equality. It then employs a transformation loop, guided by a search that uses TED as heuristic function, to find and normalize the minimal achievable distance.

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; Poiroux et al., 2024) between formal statements emerges as a widely accepted standard for automated evaluation, but its effectiveness is constrained by the current progress in automated theorem proving.

Operator Trees. Operator Trees (OPTs) represent mathematical expressions as syntax trees, with operators as internal nodes and operands as leaves (Zanibbi et al., 2002). Compared with sequence-based formula representations, OPTs explicitly capture the hierarchical structure and semantic relations within expressions, preserving operator precedence and operand dependencies (Zanibbi & Blostein, 2012; Hu et al., 2013). These properties have made OPTs foundational to applications in mathematical information retrieval (MIR). For instance, systems extract structural features from OPTs for similarity searching (Zhong & Zanibbi, 2019) or combine them with other representations for formula retrieval, as in the MCAT system (Kristianto et al., 2016). The utility of OPTs also extends to deep learning, where models like MathBERT (Peng et al., 2021) integrate OPT structures during pretraining to enhance semantic understanding , and encoders such as FORTE (Wang et al., 2021) learn formula representations directly from OPT-based inputs.

3 METHODOLOGY

This section details our proposed framework ASSESS, which is illustrated in Figure 1. We first establish the theoretical foundations in Section 3.1. We then present the two core metrics: an initial evaluation using a standard Tree Edit Distance (TED) Similarity (Section 3.2), followed by our primary contribution, the semantically-enhanced TransTED Similarity metric (Section 3.3).

3.1 Preliminaries

Our framework quantifies statement similarity by computing the distance between their structural representations. We model these representations as vertices within a weighted, undirected graph, where dissimilarity is captured by the shortest-path distance between them. To formalize this graph-based approach, we begin with the definition of a pseudometric space.

A **pseudometric space** is a set X equipped with a function $d : X \times X \to \mathbb{R}_{\geq 0}$, called a pseudometric, that satisfies the following axioms for all $x, y, z \in X$:

- Identity: $d(x, x) = 0$
- Symmetry: $d(x, y) = d(y, x)$
- Triangle inequality: $d(x, z) \leq d(x, y) + d(y, z)$

Crucially, a pseudometric differs from a metric in that $d(x, y) = 0$ does not necessarily imply $x = y$. This property is essential for our application, as structurally distinct formal statements can be semantically equivalent (e.g., $a + b$ and $b + a$). For generality, we allow the codomain to be the extended non-negative real numbers $[0, \infty]$, noting that the operation $\infty - \infty$ is undefined.

Definition 1 (Shortest-path distance)**.** *Let $G = (V, E, w)$ be an undirected, weighted graph, where V is the set of vertices, E is the set of edges, and $w : E \to [0, \infty]$ is the edge weight function. The shortest path distance $d : V \times V \to [0, \infty]$ is defined as:*

$$d(u, v) := \inf \left\{ \sum_{e \in p} w(e) \mid p \textit{ is a path from } u \textit{ to } v \textit{ with finitely many edges} \right\}$$

By convention, $d(v, v) = 0$. If no path with finitely many edges exists between u and v, the set of paths is empty, and the infimum over an empty set is ∞, so $d(u, v) = \infty$. This function d satisfies the required axioms, confirming that (V, d) constitutes a pseudometric space.

3.2 TED Similarity

Our baseline metric, TED Similarity, quantifies structural correspondence between two formal statements. This is achieved in three steps: (1) representing each statement as a hierarchical Operator Tree (OPT); (2) computing the Tree Edit Distance (TED) between the two OPTs; and (3) normalizing this distance to produce a final similarity score.

3.2.1 OPT Construction

To capture the hierarchical structure of formal statements, we represent each as a labeled, ordered Operator Tree (OPT). We leverage the Lean Language Server to parse a statement, from which we derive the tree's topology based on the nested scopes of its elements: operators become parent nodes and their arguments become ordered children. During construction, we apply two additional transformations to standardize the tree structure:

- **Placeholder Representation:** We append a placeholder `<SLOT>` to the labels of all non-leaf (operator) nodes. This explicitly marks the node's functional role, disambiguating operators from operands that might share the same name.
- **Parentheses Omission:** Parentheses enforce operator precedence in a linear string representation. Since a tree's topology inherently encodes this hierarchy, parentheses are redundant structural artifacts and are omitted from the OPT.

As an example, Figure 2 demonstrates the constructed OPT for the following formal statement:

```
theorem eq : ((Σ x ∈ Finset.range 10, (x + 1) ^ 2) % 10 = 5) = ((Σ
 n ∈ Finset.Icc 1 9, n ^ 2) % 10 = 5) := by sorry
```

3.2.2 Metric Calculation

With formal statements converted to OPTs, we can instantiate the pseudometric space defined in Section 3.1. The set of vertices X becomes the set of all possible OPTs, and the distance function d is the Tree Edit Distance (TED).

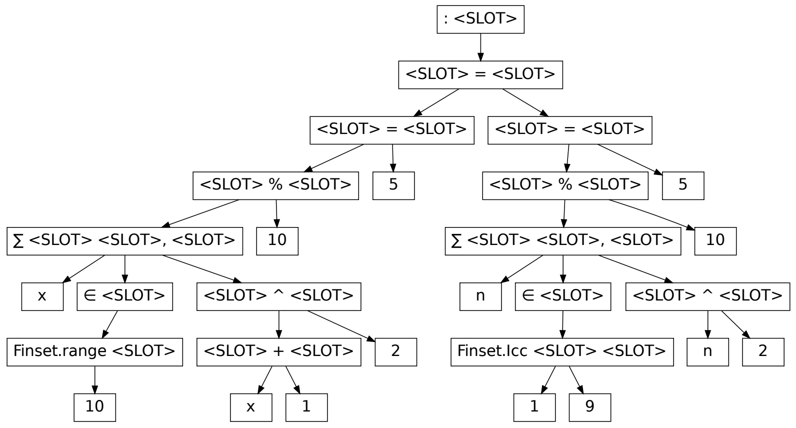

Figure 2: The Operator Tree (OPT) of a formal statement.

Definition 2 (Tree Edit Distance). *Let X be the set of all labeled, ordered Operator Trees (OPTs). The Tree Edit Distance (TED), $d_{TED} : X \times X \to \mathbb{R}_{\geq 0}$, between two trees $T_1, T_2 \in X$ is the minimum total cost of a sequence of operations that transforms T_1 into T_2. The standard operations and their associated non-negative costs are:*

- ***Delete** a node, connecting its children to its parent in order. Cost: c_{del}.*
- ***Insert** a node, the inverse of a delete operation. Cost: c_{ins}.*
- ***Relabel** a node by changing its label. Cost: c_{rel}.*

For d_{TED} to be a valid pseudometric, the costs must satisfy $c_{del} = c_{ins}$, ensuring symmetry.

To enable meaningful comparisons across trees of varying sizes, we define TED Similarity by normalizing the absolute Tree Edit Distance (TED) relative to tree size.

Definition 3 (TED Similarity). *The TED Similarity between two OPTs, T_1 and T_2, is:*

$$\mathrm{sim}_{TED}(T_1, T_2) := 1 - \frac{d_{TED}(T_1, T_2)}{\max\{|T_1|, |T_2|\}},$$

where $|T|$ is the number of nodes in tree T.

This formulation assumes unit costs for all operations ($c_{\text{del}} = c_{\text{ins}} = c_{\text{rel}} = 1$), a standard convention we adopt (Zhang & Shasha, 1989). The denominator $\max\{|T_1|, |T_2|\}$ serves as a normalization factor, representing the cost of deleting all nodes in the larger tree. This ensures the similarity score is bounded in $[0, 1]$, where 1 indicates identical trees.

3.3 TransTED Similarity

While TED Similarity effectively captures structural differences, it is syntactically rigid and penalizes semantically equivalent expressions, such as $a + b$ and $b + a$. To address this, we introduce TransTED Similarity, which augments the TED with semantic awareness through transformations.

3.3.1 Theoretical Framework

In our framework, we define a concept, transformation, to integrate semantic awareness into distance functions. We start with an example: $i = j$ implies that $f(i) = f(j)$. In other words, $f(i) = f(j)$ is no stronger than $i = j$, because to show $f(i) = f(j)$ suffices to show $i = j$. We say that the comparison of $f(i)$ and $f(j)$ can be transformed to the comparison of i and j. Generally, a weaker comparison can be transformed to a stronger one. Therefore, we hope that the distance between OPTs of the transformed comparison is no smaller than that between OPTs of the original one.

Formally, let X be the set of all Operator Trees (OPTs). We define a new pseudometric, d^*, on X that is constrained by the following two properties:

- Domination by TED: For any two trees $T_1, T_2 \in X$, the new distance is bounded by the original Tree Edit Distance: $d^*(T_1, T_2) \leq d_{\text{TED}}(T_1, T_2)$.
- Monotonicity under Transformation: If a pair of formal expressions (e_x, e_y) can be transformed into a logically stronger pair (e_u, e_v) (i.e., if $e_u = e_v$ implies $e_x = e_y$), then the metric must satisfy $d^*(\mathrm{OPT}(e_x), \mathrm{OPT}(e_y)) \leq d^*(\mathrm{OPT}(e_u), \mathrm{OPT}(e_v))$.

Finding the largest pseudometric d^* that satisfies these conditions can be formulated as a linear programming problem. The following theorem establishes that this problem has a unique optimal solution, which we define as **TransTED**. The full proof is provided in Appendix B.

Theorem 1. *Let X be an arbitrary set, $b : X \times X \to [0, \infty]$ be a function and T is a subset of $(X \times X)^2$. Consider the function space*

$$\mathcal{F} := \left\{ f \,\middle|\, f \text{ is a pseudometric on } X, \text{ and } \begin{cases} \forall x, y \in X, f(x,y) \leq b(x,y), \\ \forall ((x,y),(u,v)) \in T, f(x,y) \leq f(u,v). \end{cases} \right\}.$$

Then there exists a unique maximum function $\overline{f} \in \mathcal{F}$ such that for any $(x, y) \in X \times X$,

$$\overline{f}(x,y) = \sup_{f \in \mathcal{F}} f(x,y).$$

3.3.2 Implementation

Since any practical implementation is limited to a finite list of transformations, our algorithm computes a tractable upper bound of the TransTED, rather than its exact theoretical value. However, if this finite sequence of transformations proves that two expressions are semantically equivalent, the computed distance is exactly 0 - the true value, not an approximation.

Table 1: Additional Tactic Commands

Tactic command	Original Goal	Transformed Goal
`apply congrArg`	`f x = f y`	`x = y`
`apply congrFun`	`f x = g x`	`f = g`
`apply forall_congr; intro _`	`(a → b) = (a → c)`	`(_ : a) ⊢ b = c`
`apply implies_congr; all_goals (try exact?)` [1]	`(a → c) = (b → d)`	`a = b` or `c = d` or None
`ext`	`(fun (x : A) => f x) = (fun (y : A) => g y)`	`(x : A) ⊢ f x = g x`
`rw [propext and_imp]`	`a ∧ b → c`	`a → b → c`
`norm_cast`[2]	-	-

In the context of the Lean theorem prover, transformation can be conceptualized as the set of all tactics that adhere to our theoretical framework (see Theorem 1). For implementation, on the one hand, we curated a subset of these tactics, selected for their practical efficacy and detailed in Table 1. On the other hand, since these tactics operate on single equalities rather than pairs of statements, we first construct a unified expression for comparison. Given a pair of formal statements, we achieve this by forming an equality between their respective types. This process converts the statement pair into a single equation, which then serves as the input for our transformation tactics. Some concrete examples are provided in Appendix D.2.

[1]At least one of `a = b` and `c = d` should be proved by `exact?`, or otherwise the tactic command generates two new goals, which is forbidden in our heuristic search.

[2]`norm_cast` undertakes simple type coercions.

To circumvent the combinatorial explosion of applying all tactics, we apply a search algorithm. The algorithm performs a heuristic search through transformations. The guiding heuristic prioritizes tactics that reduce the TED between OPTs on the left- and right-hand sides of the equation, effectively pruning less promising branches. This approach allows our method to efficiently find short-distance transformation paths and the complete algorithm is detailed in Appendix C. Finally, to enhance interpretability, we normalize the raw TransTED distance into a TransTED Similarity score, defined in an analogous way as Definition 3.

4 Experiments

4.1 The EPLA Benchmark

We introduce **EPLA** (*Evaluating Provability and Likeness for Autoformalization*), a new benchmark designed to overcome the limitations of current autoformalization evaluations. Existing benchmarks often rely on coarse binary labels (e.g., True/False), which prevent a nuanced assessment of a metric's performance. For instance, a slightly incorrect but easily fixable translation is more valuable than one that is completely wrong, yet both would be identically classified as False. EPLA provides a more granular, versatile, and interpretable evaluation to address this critical gap.

EPLA is built upon two established datasets: miniF2F-test (Zheng et al., 2022) and ProofNet-test (Azerbayev et al., 2023). We utilize the specific versions of these datasets provided by Numina[3] for miniF2F-test and DeepSeek[4] for ProofNet-test. To generate candidate formalizations, we translate the natural language statements from these source datasets into Lean 4 using two distinct translators: the Herald Translator (Gao et al., 2024) and Gemini 2.5 Pro (Comanici et al., 2025). After applying a compilation filter to retain only syntactically valid statements, a panel of five experts with backgrounds in mathematics annotated and cross-validate each formal statement pair. The annotation was performed along two axes: **Semantic Provability**, defined as whether the ground-truth and predicted statements are mutually provable, and **Structural Likeness**, which measures the degree of structural likeness between them. The final EPLA benchmark, detailed in Table 2, comprises 524 such annotated data points.

Table 2: Statistics of the EPLA benchmark

	Herald Translator	Gemini 2.5 Pro	Total
EPLA-miniF2F	198	175	373
EPLA-ProofNet	89	62	151

Note: The intersection of these axes yields EPLA's five-label annotation scheme: (a) *Mutually Provable & Like*; (b) *Mutually Provable & Unlike*; (c) *Mutually Unprovable & Like*; (d) *Mutually Unprovable & Like* (where Like* denotes likeness after semantics-preserving transformations)*; and (e) *Mutually Unprovable & Unlike*. This granular classification enables flexible evaluation regimes. For instance, a strict evaluation could define success as categories (a) and (b), whereas a human-in-the-loop scenario might accept (a) through (d) as useful outputs.

4.2 Experiment Setting

Baselines. We benchmark TED Similarity and TransTED Similarity against several competing baseline methods to validate the efficacy of our proposed metric. To ensure fair and consistent evaluation across these methods, we establish specific conventions for handling theorem names. For string-based approaches (e.g., Identity Match, BLEU), we standardize theorem names in both ground truth and predicted formal statements to `thm`. For proof-based approaches (e.g., Definitional Equality, BEq), we designate the ground truth theorem names as `thm_P` and the predicted theorem names as `thm_Q`. For all other methods, theorem names remain unaltered.

- Identity Match: A predicted formal statement is considered correct if, after removing all whitespace, it is identical to the ground truth.

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

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

Table 3: **Overall results of the competing baselines and our metrics**. The boldface refers to the highest score and the underline indicates the next best result of the metrics. Detailed results about the number of TP, TN, FP and FN are available in Appendix D.1.

Metric	**EPLA-miniF2F**				**EPLA-ProofNet**			
	Precision	Recall	Accuracy	Kappa	Precision	Recall	Accuracy	Kappa
(Baselines)								
Identity Match	**100.00%**	9.78%	33.24%	0.05	**100.00%**	1.25%	47.68%	0.01
Typecheck	73.99%	**100.00%**	<u>73.99%</u>	0.00	52.98%	**100.00%**	52.98%	0.00
BLEU	89.69%	63.04%	67.29%	0.33	64.21%	76.25%	64.90%	0.29
Majority Voting	89.95%	61.59%	66.49%	0.33	72.86%	63.75%	<u>68.21%</u>	<u>0.37</u>
Definitional Equality	<u>98.92%</u>	33.33%	50.40%	0.20	<u>80.00%</u>	10.00%	50.99%	0.07
BEq	**100.00%**	48.91%	62.20%	0.33	**100.00%**	28.75%	62.25%	0.28
(Ours)								
TED Similarity	85.83%	74.64%	72.12%	<u>0.35</u>	65.98%	80.00%	67.55%	0.34
TransTED Similarity	86.08%	<u>85.14%</u>	**78.82%**	**0.46**	67.31%	<u>87.50%</u>	**70.86%**	**0.40**

- Typecheck: A predicted formal statement is deemed correct if it successfully compiles.
- BLEU: We follow the implementation in ProofNet (Azerbayev et al., 2023).
- Majority Voting: Following Lean Workbook (Ying et al., 2024a), we employ DeepSeek-V3.1 (Liu et al., 2024) with temperature 0.7 for 16 rounds of majority voting.
- Definitional Equality (Liu et al., 2025a): A predicted formal statement is considered correct under this metric if `example: thm_P = thm_Q := by rfl` succeeds.
- BEq (Liu et al., 2025a): This metric assesses the mutual provability between a ground-truth formal statement `thm_P` and a predicted formal statement `thm_Q`. It employs InternLM2-Math-Plus-20B (Ying et al., 2024b) with an expanded set of tactics to attempt proofs in both directions. The prediction is considered correct only if both proofs are successful.

Implementation. To ensure a fair comparison against baselines that require binary labels, we established a unified evaluation protocol. First, we map EPLA's granular annotations to a binary ground truth, designating categories (a) and (b) as True and all others as False. Second, for continuous metrics (BLEU, TED Similarity, and TransTED Similarity), we convert their scores to binary predictions by applying a threshold and the best results over all possible thresholds are reported.

All experiments were conducted on our EPLA benchmark using the Lean toolchain `v4.9.0-rc1`. With the sole exception of the GPU-dependent BEq baseline, all evaluated metrics were executed on CPUs. Our proposed metric is particularly lightweight, requiring only interaction with the Lean Language Server and REPL, thus imposing lower computational overhead.

4.3 Experiment Results

Overall Comparison. Table 3 presents a comprehensive overview of our proposed **TransTED Similarity** compared to various baseline metrics across the EPLA-miniF2F and EPLA-ProofNet benchmarks. Our primary evaluation focuses on accuracy and the Cohen's Kappa, as accuracy directly reflects the overall correctness of the evaluation, and Kappa (Cohen, 1960) offers a robust measure of agreement beyond chance. The results demonstrate that TransTED Similarity achieves state-of-the-art performance across both datasets. On EPLA-miniF2F, it secures the highest **accuracy (78.82%)** and **Kappa (0.46)**. It maintains this superior performance on the EPLA-ProofNet dataset, again leading with the highest **accuracy (70.86%)** and **Kappa (0.40)**. This consistent lead, particularly in the Kappa metric, highlights our method's robust ability to align with ground-truth judgments, outperforming all evaluated baselines.

Comparison with Baselines. As detailed in Table 3, **TransTED Similarity** overcomes key limitations observed across different categories of baseline methods. First, while proof-based metrics like **Definitional Equality** and **BEq** achieve high precision, their performance is hampered by low

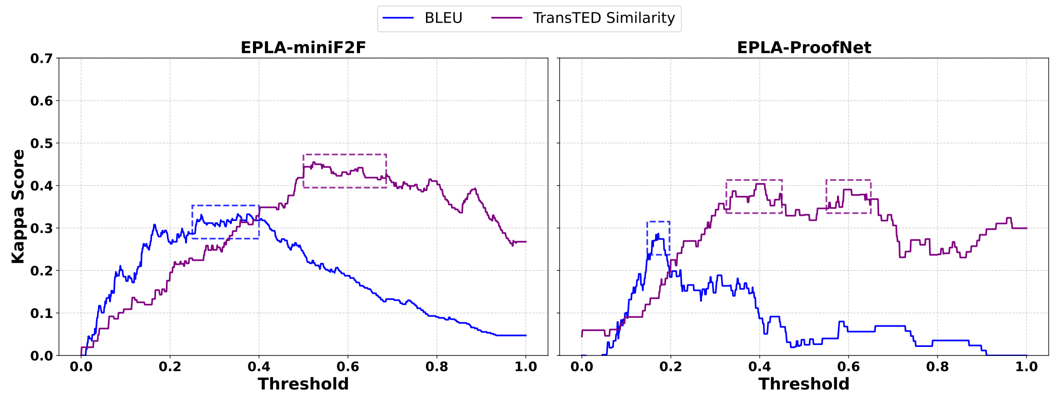

Figure 3: Comparison of BLEU and TransTED Similarity across thresholds on EPLA-miniF2F and EPLA-ProofNet. The plots illustrate that TransTED Similarity consistently achieves a higher Kappa score than BLEU and exhibits far greater stability across a wide range of decision thresholds.

recall. This brittleness stems from the inherent limitations of current Automated Theorem Provers (ATPs), which often fail to prove valid equivalences and thus produce a high rate of false negatives. Our method avoids this rigid dependency, offering a much more balanced assessment. Furthermore, TransTED Similarity significantly outperforms the LLM-based **Majority Voting** baseline, achieving superior accuracy (78.82% vs. 66.49% on EPLA-miniF2F) and a substantially higher Kappa score (0.46 vs. 0.33). Crucially, our method also sidesteps the primary drawbacks of LLM-based metrics: it is computationally inexpensive and fully reproducible. Finally, our method addresses a key weakness of n-gram-based metrics like **BLEU**: sensitivity to threshold tuning. As illustrated in Figure 3, TransTED Similarity is remarkably robust, maintaining high and stable performance across a wide range of decision thresholds. This stability is critical for practical application, as it ensures reliable outcomes without the need for costly, dataset-specific threshold optimization.

Ablation Study. We conduct an ablation study comparing our **TransTED Similarity** metric against its baseline, **TED Similarity**. The results, presented in Table 3, demonstrate that this component provides a significant performance boost on both datasets. On EPLA-miniF2F, its inclusion increases accuracy by +6.70 percentage points from 72.12% to 78.82% and raises the Kappa score from 0.35 to 0.46. This trend holds on EPLA-ProofNet, where accuracy improves by +3.31 percentage points (67.55% $\rightarrow$ 70.86%) and Kappa increases from 0.34 to 0.40. These consistent gains validate our central hypothesis: by integrating semantic transformations, our metric can successfully recognize statements that are semantically equivalent despite being syntactically diverse. This capability allows TransTED Similarity to achieve a more accurate and human-aligned evaluation where purely structural methods like the standard TED Similarity falter.

5 Conclusion

In this work, we addressed the critical gap in evaluating autoformalized statements, where existing metrics fail to adequately balance semantic and structural information. We introduced ASSESS, a novel framework whose core contribution is the TransTED Similarity metric. By augmenting the structural comparison of Operator Trees with a set of semantic transformations, our method provides a more holistic and robust measure of statement similarity. We also developed EPLA, the largest expert-annotated benchmark for this task to date. Experiments conducted on EPLA demonstrate that TransTED Similarity establishes a new state-of-the-art, outperforming existing methods in both accuracy and agreement with human judgment (Kappa). This work provides the autoformalization community with a reliable, efficient, and reproducible metric for automated evaluation.

Reproducibility Statement

We are committed to ensuring the reproducibility of our research. To this end, we provide comprehensive details of our benchmark, experimental setup, and evaluation methodology. The complete benchmark and the detailed experimental results are available in the supplementary materials. And the implementation code will be made public after the double-blind review is completed.

Specific details for reproducing our results can be found in the following sections of the paper:

- EPLA Benchmark: The construction methodology, data sources, generation process, and the complete expert annotation scheme for our benchmark are detailed in Section 4.1.
- Baselines and Experiment Setting: The Lean toolchain version, implementation details and specific configurations for all baseline methods are provided in Section 4.2 and Appendix E.
- Detailed Results: In addition to the main results in Table 3, detailed per-metric performance, including the number of True Positives, True Negatives, False Positives, and False Negatives, are available in Appendix D.1.

References

Ayush Agrawal, Siddhartha Gadgil, Navin Goyal, Ashvni Narayanan, and Anand Tadipatri. Towards a mathematics formalisation assistant using large language models. *arXiv preprint arXiv:2211.07524*, 2022.

Zhangir Azerbayev, Bartosz Piotrowski, Hailey Schoelkopf, Edward W Ayers, Dragomir Radev, and Jeremy Avigad. ProofNet: Autoformalizing and formally proving undergraduate-level mathematics. *arXiv preprint arXiv:2302.12433*, 2023.

Bruno Barras, Samuel Boutin, Cristina Cornes, Judicaël Courant, Yann Coscoy, David Delahaye, Daniel de Rauglaudre, Jean-Christophe Filliâtre, Eduardo Giménez, Hugo Herbelin, et al. The Coq proof assistant reference manual. *INRIA, version*, 6(11), 1999.

Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Garett Cunningham, Razvan C Bunescu, and David Juedes. 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.

Leonardo De Moura, Soonho Kong, Jeremy Avigad, Floris Van Doorn, and Jakob von Raumer. 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.

Guoxiong Gao, Yutong Wang, Jiedong Jiang, Qi Gao, Zihan Qin, Tianyi Xu, and Bin Dong. Herald: A natural language annotated Lean 4 dataset. In *The Thirteenth International Conference on Learning Representations*, 2024.

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

Xuan Hu, Liangcai Gao, Xiaoyan Lin, Zhi Tang, Xiaofan Lin, and Josef B Baker. Wikimirs: a mathematical information retrieval system for wikipedia. In *Proceedings of the 13th ACM/IEEE-CS joint conference on Digital libraries*, pp. 11–20, 2013.

Albert Q Jiang, Wenda Li, and Mateja Jamnik. Multilingual mathematical autoformalization. *arXiv preprint arXiv:2311.03755*, 2023.

Giovanni Yoko Kristianto, Goran Topic, and Akiko Aizawa. Mcat math retrieval system for ntcir-12 mathir task. In *NTCIR*, 2016.

Zenan Li, Yifan Wu, Zhaoyu Li, Xinming Wei, Xian Zhang, Fan Yang, and Xiaoxing Ma. Autoformalize mathematical statements by symbolic equivalence and semantic consistency. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Qi Liu, Xinhao Zheng, Xudong Lu, Qinxiang Cao, and Junchi Yan. Rethinking and improving autoformalization: Towards a faithful metric and a dependency retrieval-based approach. In *The Thirteenth International Conference on Learning Representations*, 2025a.

Xiaoyang Liu, Kangjie Bao, Jiashuo Zhang, Yunqi Liu, Yuntian Liu, Yu Chen, Yang Jiao, and Tao Luo. ATLAS: Autoformalizing theorems through lifting, augmentation, and synthesis of data. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b.

Jianqiao Lu, Yingjia Wan, Yinya Huang, Jing Xiong, Zhengying Liu, and Zhijiang Guo. Formalalign: Automated alignment evaluation for autoformalization. In *The Thirteenth International Conference on Learning Representations*, 2024a.

Jianqiao Lu, Yingjia Wan, Zhengying Liu, Yinya Huang, Jing Xiong, Chengwu Liu, Jianhao Shen, Hui Jin, Jipeng Zhang, Haiming Wang, et al. Process-driven autoformalization in Lean 4. *arXiv preprint arXiv:2406.01940*, 2024b.

Leonardo de Moura and Sebastian Ullrich. 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.

Logan Murphy, Kaiyu Yang, Jialiang Sun, Zhaoyu Li, Anima Anandkumar, and Xujie Si. Autoformalizing euclidean geometry. In *Forty-first International Conference on Machine Learning*, 2024.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 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.

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

Shuai Peng, Ke Yuan, Liangcai Gao, and Zhi Tang. Mathbert: A pre-trained model for mathematical formula understanding. *arXiv preprint arXiv:2105.00377*, 2021.

Auguste Poiroux, Gail Weiss, Viktor Kunčak, and Antoine Bosselut. Improving autoformalization using type checking. *arXiv preprint arXiv:2406.07222*, 2024.

Christian Szegedy. 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.

The mathlib Community. The Lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*. ACM, January 2020.

Qingxiang Wang, Cezary Kaliszyk, and Josef Urban. 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.

Zichao Wang, Andrew S Lan, and Richard G Baraniuk. Mathematical formula representation via tree embeddings. In *iTextbooks@ AIED*, pp. 121–133, 2021.

Ke Weng, Lun Du, Sirui Li, Wangyue Lu, Haozhe Sun, Hengyu Liu, and Tiancheng Zhang. Autoformalization in the era of large language models: A survey. *arXiv preprint arXiv:2505.23486*, 2025.

Yuhuai Wu, Albert Qiaochu Jiang, Wenda Li, Markus Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. Autoformalization with large language models. *Advances in Neural Information Processing Systems*, 35:32353–32368, 2022.

Yutong Wu, Di Huang, Ruosi Wan, Yue Peng, Shijie Shang, Chenrui Cao, Lei Qi, Rui Zhang, Zidong Du, Jie Yan, et al. Stepfun-formalizer: Unlocking the autoformalization potential of llms through knowledge-reasoning fusion. *arXiv preprint arXiv:2508.04440*, 2025.

Huaiyuan Ying, Zijian Wu, Yihan Geng, JIayu Wang, Dahua Lin, and Kai Chen. 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*, 2024a.

Huaiyuan Ying, Shuo Zhang, Linyang Li, Zhejian Zhou, Yunfan Shao, Zhaoye Fei, Yichuan Ma, Jiawei Hong, Kuikun Liu, Ziyi Wang, et al. Internlm-math: Open math large language models toward verifiable reasoning. *arXiv preprint arXiv:2402.06332*, 2024b.

Richard Zanibbi and Dorothea Blostein. Recognition and retrieval of mathematical expressions. *International Journal on Document Analysis and Recognition (IJDAR)*, 15(4):331–357, 2012.

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

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

Lan Zhang, Xin Quan, and André Freitas. 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.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. miniF2F: a cross-system benchmark for formal Olympiad-level mathematics. In *International Conference on Learning Representations*, 2022.

Wei Zhong and Richard Zanibbi. Structural similarity search for formulas using leaf-root paths in operator subtrees. In *European Conference on Information Retrieval*, pp. 116–129. Springer, 2019.

Jin Peng Zhou, Charles E Staats, Wenda Li, Christian Szegedy, Kilian Q Weinberger, and Yuhuai Wu. Don't Trust: Verify – grounding LLM quantitative reasoning with autoformalization. In *The Twelfth International Conference on Learning Representations*, 2024.

A The Use of Large Language Models (LLMs)

We utilized a large language model (LLM) as a general-purpose writing assistant during the preparation of this manuscript. The primary role of the LLM was to polish the text written by the authors, such as improving grammar. All intellectual contributions, including the research methodology, experimental results, and their interpretation, are entirely the work of the authors. The LLM's function was strictly that of an assistive tool for improving the quality of the written presentation.

B Proof of Theorem 1

Here we provide the full proof of the Theorem 1.

Proof. $0 \in \mathcal{F}$ so $\mathcal{F} \neq \emptyset$. Let $\widehat{f} : X \times X \to [0, \infty]$ be a function defined as

$$\widehat{f}(x, y) = \sup_{f \in \mathcal{F}} f(x, y).$$

It suffices to show that $\widehat{f} \in \mathcal{F}$. By definition of $\widehat{f}$, for any point pairs $(x, y) \in X \times X$ and any neighbourhood $\mathcal{N}$ of $\widehat{f}(x, y)$, there exists $f_{x,y,\mathcal{N}} \in \mathcal{F}$ such that

$$f_{x,y,\mathcal{N}}(x, y) \in \mathcal{N}.$$

Now we check that $\widehat{f}$ satisfies these properties one by one.

First, we verify that $\widehat{f}$ is a pseudometric. Take any $x, y, z \in X$. Since for any $f \in \mathcal{F}$, $f(x, x) = 0$,

$$\widehat{f}(x, x) = \sup_{f \in \mathcal{F}} f(x, x) = \sup_{f \in \mathcal{F}} 0 = 0.$$

Similarly, for any $f \in \mathcal{F}$, $f(x, y) = f(y, x)$, so

$$\widehat{f}(x, y) = \sup_{f \in \mathcal{F}} f(x, y) = \sup_{f \in \mathcal{F}} f(y, x) = \widehat{f}(y, x).$$

To show that the triangle inequality holds, we introduce an arbitrary neighbourhood $\mathcal{N}$ of $\widehat{f}(x, z)$. Then

$$\mathcal{N} \ni f_{x,z,\mathcal{N}}(x, z) \leq f_{x,z,\mathcal{N}}(x, y) + f_{x,z,\mathcal{N}}(y, z) \leq \widehat{f}(x, y) + \widehat{f}(y, z).$$

Let $\mathcal{N}$ is an arbitrary neighbourhood of $\widehat{f}(x, z)$, we have $\widehat{f}(x, z) \leq \widehat{f}(x, y) + \widehat{f}(y, z)$.

Next, we show that $\widehat{f}$ satisfies the extra requirements in the definition of $\mathcal{F}$. Since for any $f \in \mathcal{F}$, $f(x, y) \leq b(x, y)$, we have

$$\widehat{f}(x, y) \leq b(x, y).$$

Take any $((x, y), (u, v)) \in T$. Notice that for any $f \in \mathcal{F}$, $f(x, y) \leq f(u, v)$. Now for any neighbourhood $\mathcal{N}$ of $\widehat{f}(x, y)$, we have

$$\mathcal{N} \ni f_{x,y,\mathcal{N}}(x, y) \leq f_{x,y,\mathcal{N}}(u, v) \leq \widehat{f}(u, v),$$

hence $\widehat{f}(x, y) \leq \widehat{f}(u, v)$. □

C Pseudocode for TransTED

This section provides the detailed pseudocode for our TransTED algorithm. The implementation consists of two main functions: a primary wrapper `TransTED` that handles initial checks, and the core `TED AFTER TRANSFORMATION` function, which executes the heuristic search. The search `(line 13)` is guided by using the Tree Edit Distance as its heuristic `(line 18)`. If a proof of equivalence is found, the distance is 0; otherwise, if the time limit is exceeded, the algorithm returns the smallest TED found among all visited nodes `(line 21)`.

Algorithm 1 TransTED

```
 1: function TRANSTED(FL1, FL2)
 2:     eq ← (FL1 = FL2)
 3:     if Compile(eq) fails then
 4:         return TED(eq)
 5:     end if
 6:     return TEDAFTERTRANSFORMATION(eq)
 7: end function

 8: function TEDAFTERTRANSFORMATION(eq)
 9:     eq ← UseTactic(eq, norm_num1)
10:     if Completed(eq) ∨ Completed(UseTactic(eq, rfl)) then
11:         return 0
12:     end if
13:     HeuristicSearch(
14:       start : eq,
15:       stop : the goal completed ∨ time limit exceeded (TLE)
16:       search method : suggestions by rw? and some given additional tactic commands
17:       valid nodes: a single goal of equality, or "completed"
18:       heuristic function : TED between the expressions trees of both sides of the equality
19:     )
20:     if TLE then
21:         return smallest TED of all the visited nodes
22:     else
23:         return 0                          ▷ The goal is completed within the time limit
24:     end if
25: end function
```

D Experimental Details and Case Studies

D.1 Detailed Experimental Results

This section provides a detailed breakdown of the experimental results on the EPLA-miniF2F and EPLA-ProofNet datasets in Table 4 and Table 5, respectively. The tables report the number of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) for each evaluated metric.

Table 4: Detailed experimental results of automated evaluation metrics on EPLA-miniF2F.

Metric	EPLA-miniF2F							
	TP	TN	FP	FN	Precision	Recall	Accuracy	Kappa
(Baselines)								
Identity Match	27	97	0	249	**100.00%**	9.78%	33.24%	0.05
Typecheck	276	0	97	0	73.99%	**100.00%**	<u>73.99%</u>	0.00
BLEU	174	77	20	102	89.69%	63.04%	67.29%	0.33
Majority Voting	170	78	19	106	89.95%	61.59%	66.49%	0.33
Definitional Equality	92	96	1	184	<u>98.92%</u>	33.33%	50.40%	0.20
BEq	135	97	0	141	**100.00%**	48.91%	62.20%	0.33
(Ours)								
TED Similarity	206	63	34	70	85.83%	74.64%	72.12%	<u>0.35</u>
TransTED Similarity	235	59	38	41	86.08%	<u>85.14%</u>	**78.82%**	**0.46**

Table 5: Detailed experimental results of automated evaluation metrics on EPLA-ProofNet.

Metric	**EPLA-ProofNet**							
	TP	TN	FP	FN	Precision	Recall	Accuracy	Kappa
(Baselines)								
Identity Match	1	71	0	79	**100.00%**	1.25%	47.68%	0.01
Typecheck	80	0	71	0	52.98%	**100.00%**	52.98%	0.00
BLEU	61	37	34	19	64.21%	76.25%	64.90%	0.29
Majority Voting	51	52	19	29	72.86%	63.75%	68.21%	0.37
Definitional Equality	8	69	2	72	80.00%	10.00%	50.99%	0.07
BEq	23	71	0	57	**100.00%**	28.75%	62.25%	0.28
(Ours)								
TED Similarity	64	38	33	16	65.98%	80.00%	67.55%	0.34
TransTED Similarity	70	37	34	10	67.31%	87.50%	**70.86%**	**0.40**

D.2 Illustrative Examples of TransTED Similarity

This section provides a set of illustrative examples to offer a qualitative understanding of TransTED Similarity's performance, particularly in cases where TED Similarity fails. Each example is presented in a consistent format, including the original natural language statement (NL), the ground-truth formalization (Label), and the model's output (Prediction). We then present the scores from both metrics, the transformation path (Proof) discovered by our method, and a brief analysis.

# exercise_1_1b (from ProofNet)

NL: If r is rational $(r \neq 0)$ and x is irrational, prove that rx is irrational.

Label:

```
theorem exercise_1_1b (x : ℝ) (y : ℚ) (h : y ≠ 0) : (
Irrational x ) → Irrational ( x * y ) := by sorry
```

Prediction:

```
theorem mul_rat_tac_11959 (r : ℚ) (x : ℝ) (h : Irrational x) (
hr : r ≠ 0) : Irrational (r * x) := by sorry
```

TED Similarity: 0.23809523809523814
TransTED Similarity: 1

Proof given by TransTED:

```
example :
   (∀ (x : ℝ) (y : ℚ), y ≠ 0 → Irrational x → Irrational (x * ↑y
       )) =
     ∀ (r : ℚ) (x : ℝ), Irrational x → r ≠ 0 → Irrational (↑r *
         x) := by
  rw [forall_swap]
  apply forall_congr; intro _
  rw [← isRegular_iff_ne_zero']
  apply forall_congr; intro _
  rw [forall_swap]
  rw [@Rat.cast_comm] -- given by rw?
```

Comparing the label and prediction, human evaluator can easily match related variables and assumptions and give a correct judgment that they are semantically almost identical. However, TED Similarity soon gets into trouble dealing with lots of variable renaming and structure adjustment and gives out a really low similarity score. Fortunately, suggestions provided by `rw?` together with additional tactics, such as `apply forall_congr; intro _` , make it possible to complete the natural matching step by step, leading to a more accurate estimation on semantic similarity.

# mathd_algebra_142 (from miniF2F)

NL: A line ℓ passes through the points $B(7, -1)$ and $C(-1, 7)$. The equation of this line can be written in the form $y = mx + b$; compute $m + b$. Show that it is 5.

Label:
```
theorem mathd_algebra_142 (m b : ℝ) (h₀ : m * 7 + b = -1) (h₁ :
 m * -1 + b = 7) : m + b = 5 := by sorry
```

Prediction:
```
theorem my_favorite_theorem : let B : ℝ × ℝ := (7, -1); let C
: ℝ × ℝ := (-1, 7); ∀ m b : ℝ, (B.2 = m * B.1 + b ∧ C.2 = m *
C.1 + b) → m + b = 5  := by sorry
```

TED Similarity: -0.033333333333333344
TransTED Similarity: 1

Proof given by TransTED:

```
example :
(∀ (m b : ℝ), m * 7 + b = -1 → m * -1 + b = 7 → m + b = 5) =
(let B := (7, -1);
let C := (-1, 7);
∀ (m b : ℝ), B.2 = m * B.1 + b ∧ C.2 = m * C.1 + b → m + b = 5)
    := by
  apply forall_congr; intro _
  rw [mul_neg_one] -- given by rw?
  apply forall_congr; intro _
  rw [and_symm_left] -- given by rw?
  rw [and_symm_right] -- given by rw?
  rw [propext and_imp]
```

In this example, TransTED algorithm successfully matches corresponding variables and hypotheses by applying proper tactics and theorems, and gives a reliable evaluation result beyond structural likeness.

D.3 False Positives in Definitional Equality

This case study investigates the root cause of the false positives recorded by the Definitional Equality metric—notably, one on the EPLA-miniF2F and two on EPLA-ProofNet.

# mathd_algebra_176

NL: Expand the product $(x + 1)^2 \cdot x$. Show that it is $x^3 + 2x^2 + x$.

Label:
```
theorem thm_P (x : ℝ) : (x + 1) ^ 2 * x = x ^ 3 + 2 * x ^ 2 + x
 := by sorry
```

Prediction:
```
theorem thm_Q {R : Type*} [CommRing R] (x : R) :  (x + 1) ^ 2 *
 x = x ^ 3 + 2 * x ^ 2 + x  := by sorry
```

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

In the Prediction, the statement was generalized by introducing an implicit type variable $\{R :$ `Type*`$\}$ with a typeclass assumption `[CommRing` R`]` instead of the concrete type $\mathbb{R}$, resulting in a semantic difference; however, Lean will automatically instantiate $R := \mathbb{R}$ and resolve the

`CommRing` instance, so in Definitional Equality the compiler infers these instantiations and the two declarations become definitionally equal, thus passing the verifier and producing a false positive.

# 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 the Prediction, the model introduced an implicit type variable $\{\Omega$: `Type u_1`$\}$ with a typeclass assumption `[Infinite Ω]` instead of the concrete type $\mathbb{N}$; however, Lean can instantiate $\Omega := \mathbb{N}$ and use the existing `Infinite ℕ` instance, so in Definitional Equality the compiler infers these instantiations and the two declarations become definitionally equal, thus passing the verifier and producing 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.

E PROMPT TEMPLATES

This section presents the prompts for the majority voting phase, which uses InternLM2-Math-Plus-7B (Ying et al., 2024b) for back-translation and DeepSeek-V3.1 (Liu et al., 2024) for NLI Check.

Prompt Template for Back-Translation

[UNUSED_TOKEN_146]user\nConvert the formal statement into natural language:\n"‘ lean\nformal_statement\n"’[UNUSED_TOKEN_145]\n[UNUSED_TOKEN_146]assistant\n

Prompt Template for NLI Check

Please check following two math problems is same or different? Please consider each statement in two problems, they are different if any statement is different. Please point out any differences you found. Please reply **same** or **different** in the final sentence with bold format.
Problem 1: {THM_1}
Problem 2: {THM_2}