A Similarity Metric for Evaluating Natural Language to Logic Translation

Anonymous ACL submission

Abstract

To introduce human-created conventions specified in natural language into multi-agent systems (MASs), it is necessary to convert them into rules that agents can process and reason about. In this paper we introduce a quantifier-free higher-order formal language for representing these conventions. To facilitate the translation, we employ large language models (LLMs) to convert natural language conventions into this formal logic. However, assessing the quality of these automatically generated translations requires similarity metrics that capture semantic meaning, while faithfully reflecting differences in logical structure. Existing metrics used in natural language to first-order logic (NL-FOL) translation tasks, such as BLEU and Logical Equivalence, prove insufficient for this purpose. They usually lack sensitivity to semantic similarities between conceptually related predicates and fail to account for structural differences between formulae. This paper proposes a novel similarity metric designed specifically to address these limitations by accounting for similarity across both semantic and structural dimensions. We test our proposed metric and compare it to BLUE and LE on the NL-FOL pairs from the MALLS dataset and show that it provides more comprehensive and reliable similarity assessments.

1 Introduction

Conventions can be defined as recurrent behaviour patterns of communities (Balke et al., 2013) that increase the predictability of interaction outcomes. The term 'convention' is often related to patterns resulting from an agreement among members of a given community or culture. The term 'norm' is often associated with legal aspects of behaviour and contains rewards or sanctions. In this paper, we will use the term 'convention', but our proposal could be applied to norms or rules, as they all share the same basic structure. In an AI context, such conventions can coordinate agents' actions in multi-agent systems (MASs) and simplify decision-making machinery. In general, not all agents are necessarily willing or capable of adhering to the same conventions. However, in a cooperative multi-agent system, it is reasonable to assume that agents will align on shared conventions to enhance coordination and overall system performance.

When these conventions are expressed in natural language (NL), they must first be translated into a formal representation that agents can process. Large language models (LLMs) have shown strong capability in performing such translations. However, reliable methods are still needed to assess the quality of such translations. To enable systematic evaluation, we adopt gold-standard formalisations of NL conventions as reference standards.

Evaluating the similarity between formal representations requires sensitivity to both logical structure and semantic meaning. Existing metrics adapted from natural language to first-order logic (NL-FOL) translation, such as BLEU and Logical Equivalence (LE), fall short in this regard. For instance, BLEU measures surface token overlap but ignores semantic relationships between predicates, while LE captures truth-table similarity but overlooks finer structural and semantic nuances. Consequently, neither provides a comprehensive picture of how closely a generated formula aligns with the intended meaning.

To overcome these limitations, we propose a new similarity metric that compares formulae by converting them into Disjunctive Normal Form (DNF)-like trees and measuring similarity across both semantic and structural dimensions. We then benchmark our metric against BLEU and LE, examining not only correlations but also cases of disagreement. Our analysis reveals that the proposed similarity correlates moderately with both baselines, while offering deeper insights in situations where BLEU and LE diverge. These results highlight the

advantages of combining semantic and structural perspectives, and they underscore the limitations of existing metrics when applied in isolation. Our proposed similarity metric is designed to work with quantifier-free higher-order logic (HOL). This formalism was chosen because it provides the expressive power needed to capture complex convention structures while avoiding the computational overhead introduced by quantifiers.

The contributions of this work are as follows: (1) we introduce a quantifier-free formal language and tree-based representation that enables tractable yet comprehensive similarity comparison; (2) we develop and validate a composite similarity metric that integrates semantic and structural assessment; and (3) we provide detailed analysis of metric behaviours, revealing specific limitations of the different approaches and identifying directions for future improvement in formal representation evaluation.

The remainder of the paper is organised as follows. Section 2 reviews related work, and Section 3 introduces our quantifier-free HOL. Section 4 details the proposed similarity metric, while Section 5 and Section 6 present and analyse the experimental results. Section 7 discusses the strengths and limitations of our approach, before concluding with Section 8.

2 Background

Some NLP techniques for norm extraction have been presented and evaluated in Ferraro et al. (2020), including relation extraction and semantic parsing. Nevertheless, they did not involve LLMs in their work. In this section, we focus on the techniques that use LLMs for norm extraction. Some previous works applied LLMs to extract implicit norms as natural language text from sources that were themselves written in natural language (Fung et al., 2023; Qu et al., 2024; Zin et al., 2024; Pujari and Goldwasser, 2025). Apart from these, other works required the LLMs learn norm from input and follow them in their downstream tasks, but did not require them to generate the norms as output (Bachinger et al., 2024; Kim et al., 2025). Haque and Singh (2024) proposed an approach for formalising norms by extracting normative relationships from unstructured legal contracts via LLMs. Similarly, Janatian et al. (2023) aimed to create structured representations for legalisation. One of the most recent symbolic attempts is to generate Defeasible Deontic Logic (DDL) from nature language description (Horner et al., 2025). The authors propose thorough criteria to compare the generated DDL with expert-crafted formalisation from different aspects, including completeness, syntactic and semantic correctness, deontic modality accuracy, precondition appropriateness, and meaningfulness/reuse of atom names. Each aspect is represented as binary value (1 for satisfied, 0 for unsatisfied). Although the calculation of final success score is formally defined, they did not clarify if they automated the procedure for deciding if an aspect is satisfied or not. Moreover, this approach is tailored specifically to DDL, as it is a non-monotonic logic that incorporates features absent in monotonic logics such as First-Order logic (FOL). As a result, adapting it to our setting is challenging.

In our search for suitable similarity metrics, we extend our focus to NL–FOL translations, since our proposed language remains closely related to FOL and this area has been more extensively studied. However, we have found that many works are focusing on the downstream task accuracy (e.g., inference) (Olausson et al., 2023; Tian et al., 2021; Faghihi et al., 2024; Han et al., 2024; Raheja et al., 2024), and they do not include metrics for similarities between FOLs. Some works also applied solvers or verifiers to validate if two FOLs are logically equivalent (Thatikonda et al., 2024; Wang et al., 2024; Karia et al., 2025; Ryu et al., 2025). In Yang et al. (2024) they adapted BLEU and LE (also used in (Xu et al., 2024; Liu, 2025)), which are suitable for our language. For BLEU calculation, a tokenizer is used to convert FOLs into tokens, including operators, terms, parentheses and commas. BLEU measures similarity by calculating n-gram overlap between token sequences of generated and gold standard FOLs. As for LE, a parser first parses FOLs as CFG trees based on token sequences. Predicates from generated FOLs are then bound to gold standard ones, and LE calculates the overlap ratio between their truth tables. Therefore, we have chosen to use them as our baseline metrics.

3 A Quantifier-free Higher-order Logic

Since machine-readable rules for conventions are highly similar to FOL in structure, and there are more established models and datasets for NL-FOL translation, we initially decided to use FOL for representation.

For instance, consider this rule for the convention ‘playing chop card’. It is from the board game

Hanabi, and this rule reads as follows: 'If a player played a card from their chop slot (which is by default the right-most in a player's hand, where the card is unknown to this player), then they are considered to have played their chop card'.

$$
\begin{array}{ll}
convention(& \\
play\_card(S), & \#action \\
role(Ag, Role) \ \& \ chop(Role, S), & \#condition \\
interp(Ag, play\_chop(S)) & \#interp \\
). &
\end{array}
$$

This rule can simply be broken into three parts that describe the `action`, `condition`, and `interp` (short for 'interpretation'). The agent can take the `action` when all the beliefs in `condition` are satisfied. The `action` taken under specific `condition` should be interpreted as a conventional move by the other agents, marked in `interp`. This rule can be written in a FOL implication-style clause:

$$
\begin{array}{l}
(role(Ag, Role) \wedge chop(Role, S)) \rightarrow \\
(play\_card(S) \wedge interp(Ag, play\_chop(S)))
\end{array}
$$

However, it quickly became evident that FOL is not enough. Theory of mind and nested beliefs are common in gaming conventions, like Hanabi's. Foe example, a nested belief 'Alex believes that Sam believes that they have played their chop card' would be represented as `Believe(Alex,Believe(Sam,done(Alex,play_chop),turn))`. This is a higher-order predicate in logic.

Therefore, we propose a quantifier-free HOL. In other words, while we allow nested predicates, we do not include quantifiers for the time being. We also note that while we do have predicates such as `Believe`, we are not truly stepping into modal logic since we do not (in this work) deal with such predicates as operators.

For evaluating the similarity of formulae, we convert each formula in our language into a DNF-like tree so we can compare those trees. This conversion is possible in our proposed language because not having quantifiers allows us to avoid the infinite branching issues that arise in first or higher order logics. Furthermore, DNF-style distributivity of 'and' over 'or' still applies. Though the literals are no longer flat predicates but possibly nested predicate terms. To create those nested predicate terms, we treat each predicate symbol as a node in the tree, whose arguments become its children. Just like a DNF, disjunctions will be at the top of the tree, conjunctions a level below that, and the nested predicate atoms as leaves. In summary, the only difference from classical DNF is that the literals are not atomic variables but structured/nested predicate terms.

4 The Semantic Similarity Metric

Assuming we have a well-formed formula in our quantifier-free HOL, we represent it structurally as a DNF-like tree. This allows us to compare the semantic similarities of formulae while taking their structural composition into account, rather than considering terms in isolation. In this DNF-like representation, each predicate symbol is treated as a node in the tree, and its arguments appear as child nodes. Nested predicates are handled naturally in this framework, since the output of one predicate node can itself serve as the child of another. Logical connectives ($\wedge$, $\vee$, $\neg$) are also represented as internal nodes in the tree.

For example, suppose we have a DNF-like formula in our higher-order setting: `(R(w,v) ∧ ¬S(i,j)) ∨ P(x,Q(y,z))`. This mirrors the disjunctive normal form structure of propositional logic, but the literals here are predicate expressions (possibly nested), rather than atomic propositional variables. The DNF-like tree structure is illustrated in Figure 1 with the tree always starting with OR as root, since it is the primary connective operator between groups of predicates. The position of the other elements in the tree, from shallow to deep, is: AND, NOT, predicate terms, and arguments.

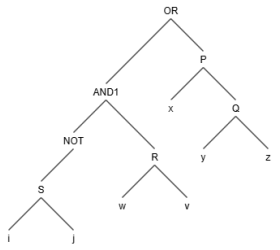

Figure 1: A example of a DNF-like tree.

We assign unique names to the AND operators in each tree to indicate that they connect distinct groups of nodes. Without this naming, information about the grouping along paths would be lost during the comparison of formulae.

After converting a formula in our quantifier-free HOL to a DNF-like tree, the similarity between two formualae in our logic becomes the similarity between the two DNF-like trees.

Suppose we are comparing two DNF-like trees, $T1$ and $T2$. To do so, and because the similarity between two trees ($TreeSim$) is asymmetric, we choose to be cautious and take the minimal similarity of each possible directional combination. This ensures the reported similarity reflects the worst-case alignment between the trees, accounting for differences in structure and grouping of subtrees:

$$Sim(T1, T2) = \min(TreeSim(T1, T2), TreeSim(T2, T1)) \quad (1)$$

This Sim has the range [0, 1]. The value closer to 1 means the trees are more similar to each other (1 for identical trees), while the value closer to 0 means less similar. For tree similarity in one direction, we first calculate the similarity between each path in $T1$ with $T2$ as a whole, and then take the average across all paths of $T1$. Intuitively, this ensures that every path in $T1$ contributes to the final score, and prevents the result from being dominated by a single path match. This similarity from $T1$ to $T2$ is defined as follows:

$$TreeSim(T1, T2) = \frac{\sum_{i=1}^{NoP(T1)} PathTreeSim(\mathrm{P}_i^{T1}, T2)}{NoP(T1)} \quad (2)$$

where $NoP(T1)$ is the number of paths in tree $T1$, and P_i^{T1} is the i-th path of $T1$.

To calculate the similarity between a single path in $T1$ and a tree $T2$, we take the maximal similarity between $T1$ and the paths in $T2$. In other words, we consider for $T1$ its best matching path in $T2$. We also consider how many times a certain path in $T2$ (P_j^{T2}) is used as the best match for the different paths in $T1$. The intuition is that if a path in $T2$ is reused to match multiple paths in $T1$, we penalise for this repetition by dividing the path similarity with $T2$ by the number of times the matching path in $T2$ has been reused. The similarity between a path in $T1$ and $T2$ is defined as follows:

$$PathTreeSim(\mathrm{P}_i^{T1}, T2) = \frac{\max_{j \in [1, NoP(T2)]} PathSim(\mathrm{P}_i^{T1}, \mathrm{P}_j^{T2})}{PathReuse(j^*(i))} \quad (3)$$

where $j^*(i)$ is the best-matching path in tree $T2$ for path P_i^{T1}, and it is defined accordingly:

$$j^*(i) = \arg \max_{j \in [1, NoP(T2)]} PathSim(P_i^{T1}, P_j^{T2})$$

And the number of times that matched path ($j^*(i)$) has also matched other paths of tree $T1$ is defined by the $PathReuse$ function accordingly:

$$PathReuse(j) = \sum_{k=1}^{NoP(T1)} \delta(j^*(k) = j)$$

where $\delta(condition)$ simply returns 1 every time the condition is true.

To calculate the similarity between two paths, the basic idea is to calculate the similarity between the nodes of each level of the path, one at a time, and take the average of those node similarities. The intuition here is that two paths are considered similar if their nodes match well across most levels. The average similarity captures the overall alignment of their structure. However, there are other penalisations that we ultimately have to introduce, and which are captured in the path similarity equation defined below:

$$PathSim(\mathrm{P}_i, \mathrm{P}_j) = \frac{\sum_{n=1}^{X} PenNodeSim(\mathrm{A}_n^{\mathrm{P}_i}, \mathrm{A}_n^{\mathrm{P}_j})}{X \cdot Harmonic\_Sum(Y)} \quad (4)$$

First, we have $X = \min(PathLength(\mathrm{P}_i), PathLength(\mathrm{P}_j))$, where $PathLength(P)$ returns the length of path P. In other words, X describes the length of the shorter path, and the averaging of node similarities stops once all nodes in the shorter path have been compared.

Now note that in Equation 4, in addition to computing the average of node similarities by dividing their summation by X, we also divide that final path similarity by the harmonic sum of Y, where $Y = (\max(PathLength(\mathrm{P}_i), PathLength(\mathrm{P}_j)) - X) + 1$, and it represents the difference between the paths' lengths. We add 1 to avoid dividing by 0 when the difference is 0. The intuition behind dividing by the harmonic sum is to penalise for differences in path lengths that result in ignoring nodes in the longer path. The choice of the harmonic sum has been motivated by its smooth penalisation effect that grows with larger differences, while avoiding overly harsh reductions. Though other penalisation strategies can be investigated in future work.

Finally, we note that the node similarity used in Equation 4 is already a penalised version of the actual similarity between two nodes ($NodeSim$).

This penalisation depends on the number of nodes whose similarity is being aggregated (X). For example, when comparing only two nodes, a simple average can be misleading. If one similarity is very high and the other very low, the high value can dominate, masking the poor alignment of the low-similarity node. As an illustration, consider the two-length paths describing 'cold(Alex)' and 'eating(Alex)'. Since both paths share the same constant at the second node ('Alex'), this node contributes a high similarity to the overall path similarity. However, the first nodes ('cold' vs. 'eating') are semantically very different, with a low similarity. To address this issue, and motivated by the initial experimental results, we decided to penalise the similarities, where lower similarities get a stronger penalisation. Furthermore, we reduce the penalisation as the number of nodes (X) grows. This is because as the number of nodes increases, the influence of individual nodes is diluted by the other nodes, so less penalisation is needed. This design ensures that the metric remains sensitive to mismatches in small sets while remaining stable for longer paths.

To achieve this, we define $PenNodeSim$ as follows:

$$PenNodeSim(\mathrm{A}_n^{\mathrm{P}_i}, \mathrm{A}_n^{\mathrm{P}_j}) = \begin{cases} NodeSim(\mathrm{A}_n^{\mathrm{P}_i}, \mathrm{A}_n^{\mathrm{P}_j}) & \text{, if} \quad X = 1 \\ NodeSim(\mathrm{A}_n^{\mathrm{P}_i}, \mathrm{A}_n^{\mathrm{P}_j})^{1+\frac{\alpha}{X}} & \text{, otherwise} \end{cases} \quad (5)$$

where X is the length of the shorter path (as defined earlier). The first case states that if we are only aggregating one node similarity, no penalisation is applied. Otherwise, the penalisation depends on the number of nodes being considered (X) and a scaling factor α, which controls the strength of the penalisation. When α is 0, no penalisation is applied. But our experiments reveal that performance is best when α is 3, 4 or 5 (we use 5 in our experiments). The larger α is, or the lower X is, the more aggressive the penalisation.

We note that our proposal includes penalisation at different stages. Naturally, alternative penalisation approaches could be examined in future work. Similarly, while our primary aggregation method has been the average, future research may explore alternative aggregation strategies.

Next, we define the similarity between two nodes ($NodeSim$). Node similarities are derived from embeddings that capture their semantic meaning, allowing us to recognise related nodes even when their symbols differ (e.g. 'buy' and 'purchase'). But first, some rules are applied before the embedding-based similarity is calculated. For example, if one or both of the nodes is an operator, then we apply the following rules. The similarity between different operators is 0 (this is the case for the similarity between AND and NOT, recall that the OR is always a root node so it never gets compared to other nodes). The similarity between an operator and a node that is not an operator (like variables, constants, or predicate symbols) is also 0. A special case is the similarity between ANDs, which are numbered uniquely in each tree to indicate distinct groups (as explained earlier in this Section). When comparing ANDs, we consider all possible matchings between the AND nodes of the two trees, and we calculate the overall semantic similarity of the two trees for all these possible AND matchings. We then retain only the matchings that yield the highest similarity between the two trees, and we discard the rest. This is important because it ensures that the similarity score reflects the most meaningful structural alignment of the trees: only the AND groups that correspond best to each other contribute to the final similarity, while misaligned groups are naturally penalised. Accordingly, we assign a 1 to the similarity of matched ANDs and a 0.2 to the similarity of unmatched ANDs. The low value for unmatched ANDs allows us to acknowledge a minimal notion of similarity without letting them substantially influence the overall score.

When comparing two nodes that are not operators (such as variables, constants, or predicate symbols), we assess their semantic similarity using embeddings. Here 'semantic similarity' is the linguistically lexical similarity calculated based on embeddings. These embeddings are obtained from a pretrained SentenceTransformers (SBERT) model (Reimers and Gurevych, 2019). Each node is encoded using the model to obtain its embedding. For nodes with unknown atomic words or compund words (connected with - or _), the model will tokenise these nodes into sub-tokens, calculate embeddings for each sub-token separately, and combine them into one embedding. Node similarities are computed as normalised cosine similarity between embeddings, scaled to the range [0, 1].

5 Experiments

To evaluate our proposed similarity metric (Equation 1), we compared it against existing metrics on

a standard NL-FOL translation task, particularly BLEU (Papineni et al., 2002; Lin and Och, 2004) and LE (Yang et al., 2024). Although BLEU and LE were originally applied for NL-FOL translation, they provide suitable baselines for our evaluation since our quantifier-free HOL is similar to FOL.

We adopt the experimental setup of Yang et al. (2024), using the MALLS dataset and LogicLLaMA for NL-FOL translation task. MALLS provides a human-verified test set of 1,000 natural language statements paired with gold-standard FOL representations generated by GPT-4.

Our proposed similarity metric operates over a subset of our HOL. Accordingly, we filter the original test set by removing NL-FOL pairs containing unsupported operators (e.g., $\bigoplus$ and $\leftrightarrow$) and eliminate quantifiers from the FOL representations. We further exclude formulae with more than five unique conjunctions in their DNF-like trees to avoid prohibitively expensive alignment computations. After filtering, 408 pairs remain. The resulting representations are treated as gold-standard formulae and denoted as **GD**. To obtain model-generated formulae, we apply LogicLLaMA, a LLaMA-2-7B model fine-tuned on MALLS, to translate the corresponding natural language statements. We set the model on HPC, and requested 2 GPU on a NVIDIA A100 node with 64 CPU cores and 192 GB memories. For 408 pairs the translation and evaluation tasks took roughly one hour. We apply the same preprocessing steps to the generated outputs, including quantifier removal, yielding the generated representations denoted as **Gen**. Each **GD**-**Gen** pair corresponds to the same natural language statement.

5.1 Evaluation

We used two baseline metrics: BLEU and LE. According to Yang et al. (2024), LE determines logical equivalence between FOLs, while BLEU confirms that the model correctly extracted predicate names and entities from natural language input.

However, we note that LE does not guarantee actual logical equivalence, as it treats each predicate as a proposition to create finite truth tables, neglecting quantifiers and variable bindings. Since it only considers truth table overlap, two formulae with completely different meanings can still achieve nearly identical truth tables. For analysis, we calculated correlations between all three metrics and defined agreement/disagreement categories for further investigation.

6 Results

BLEU and LE show no correlation with each other (r = 0.23). This result is expected since they capture different aspects of similarity. In contrast, Sim correlates moderately with both BLEU and LE (r = 0.665 for both). This suggests that our metric captures both token-level and logical structure similarities. To validate this consistency, we conducted a systematic analysis of the results. We divided the scores into three rank-based quantiles: High (top 30%), Medium (middle 40%), and Low (bottom 30%). This stratification helps mitigate distributional effects that could distort score comparisons.

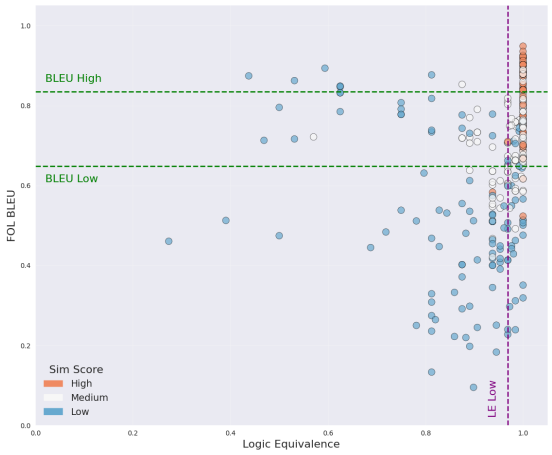

Figure 2: Sim correlation with BLEU and LE across quantiles.

Figure 2 visualises the relationship between Sim and the baseline metrics BLEU and LE. Each dot represents a single evaluated example (or case) (408 in total), plotted according to its LE score on the x-axis and BLUE score on the y-axis. The dots are coloured by their Sim quantile range (High with 123 cases, Medium with 162 cases, and Low with 123 cases), allowing visual comparison across metrics. This figure shows that Sim correlates moderately with both baseline metrics. High Sim scores occur most frequently when both BLEU and LE scores are high (about 72% of High cases). Low Sim scores are more possible to align with low BLUE and LE scores (about 54% of Low cases).

We define agreement and disagreement categories based on score ranges. Perfect agreement occurs when all three metrics fall in the same range (all high, all medium, or all low). We identify two main disagreement categories: Strong Disagreement and Mild Disagreement, each with two sub categories (marked with 'All' or score name).

Here we consider High versus Low as 'far', and High/Low versus Medium as 'close'. When both BLEU and LE are far from Sim, or only BLEU or LE is far from Sim, it falls under Strong Disagreement. For Mild Disagreement, both BLEU and LE are close to Sim, or one of them is close and the other falls in the same score range of Sim. In our analysis, we focus on Strong Disagreement cases, and the Mild Disagreement cases where BLEU and LE agree with each other. These are the cases presented in Figure 3.

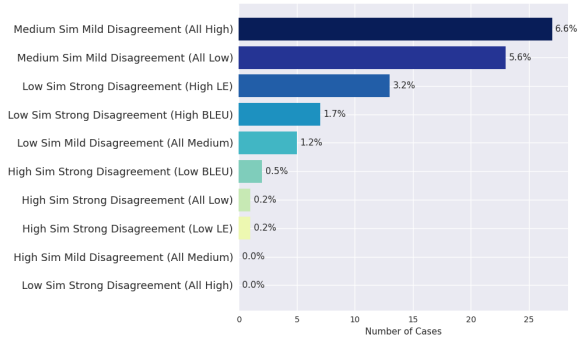

Figure 3: Overall of disagreement categories.

Results show perfect agreement occurs in 41.9% of cases, suggesting moderate level of consistency among the three metrics. An additional 23.5% of cases receive medium Sim with opposing BLUE and LE scores. This illustrates that Sim often provides intermediate assessment that bridges the discrepancy between BLEU and LE. These numbers illustrate that we have more Agreements and Mild Disagreements. In other words, disagreement is kept at reasonable levels. Strong disagreements form only 5.8% of cases. This aligns with our expectation that Sim remains consistent with other metrics. Finally, we note that, as shown in Figure 3, two disagreement categories remain empty: High Sim Mild Disagreement (All Medium), and Low Sim Strong Disagreement (All High).

For each category, we select the top 10 cases with highest disagreement scores, or all cases if the number of cases in that category is smaller than 10. For each category, we quantified the strength of disagreement between metrics. When only one baseline metric (either BLEU or LE) differed from Sim, we measured disagreement as the absolute difference between Sim and that metric. When both baseline metrics disagreed with Sim, we used the average of BLEU and LE to represent their combined behaviour and calculated the difference between Sim and this average. Details of this analysis are presented next.

6.1 Disagreements With a High Sim

Strong Disagreement cases with a high Sim are rare (0.9% in total). Disagreements occur primarily due to two factors:

BLEU assigns low scores when predicates differ lexically despite semantic similarity. Minor variations such as morphological differences (missing 's' suffix) or truncated predicate names (`CriticallyAcclaimedFilm` versus `CriticallyAcclaimed`) cause significant BLEU score drops, while Sim correctly recognises semantic similarity through embeddings.

LE is affected by how parsers handle operator precedence. Sim's parser may create different tree structures than LE's parser, particularly for expressions with complex operator combinations. For example, Sim's parser gives OR higher priority, so no matter there are parenthesis for the predicates connected by OR or not, it will offer the same parsing result, while LE's parser will be affected. See Appendix A.1 for detailed case analysis.

6.2 Disagreements with a Low Sim

Disagreements with low Sim are slightly more common (6.1%) and reveal important limitations of baseline metrics when structural differences are present.

High LE with Low Sim: LE remains insensitive to predicate argument structure. When LLMs embed arguments into predicate names (e.g., `ParticipatesInSprints(x)` instead of `ParticipatesIn(x,sprints)`), LE's edit-distance-based predicate binding overlooks these structural differences. Sim correctly penalises such cases due to different path lengths and additional tree paths from extra arguments.

High BLEU with Low Sim: High n-gram overlap can mask significant logical structure differences. Common patterns include: (1) alternative implication positions where predicates appear on opposite sides of implications, and (2) missing parentheses that change operator scope. These create large structural differences that Sim captures through DNF-like tree comparison, while BLEU's token-level analysis misses them.

Our analysis also revealed cases where embedding-based matching can misalign semantically related terms (e.g., `Romance` matching `Movie` instead of `hasRomance`), and where variable representations affect path matching. See Appendix A.2

for detailed case analysis and formula comparisons.

6.3 Disagreements with a medium Sim

Medium Sim scores with strong BLEU/LE agreement (12.2% of cases) primarily reflect Sim's calculation methodology rather than genuine metric disagreement.

All High baseline metrics: These cases involve semantically similar but imperfectly matched predicates. High overlap leads to strong baseline scores, but node mismatches (e.g., `Trails` aligned with `Mountain`, or `ContainsAlcohol` matching `NonAlcoholic`) reduce Sim. This occurs because embedding pooling from subtokens can produce vectors close in embedding space even for conceptually different or opposite terms.

All Low baseline metrics: Sim's averaging mechanism can yield medium scores when only partial differences exist. Common patterns include: (1) predicate splitting (`StudiesAndAssessesHumanBehaviorAndMentalProcesses` splits into two predicates), where one split matches well; (2) predicate merging (multiple predicates merge into compound names); and (3) extra predicates in longer representations, where additional paths have limited impact on overall similarity when other paths match well.

These cases demonstrate that Sim's bidirectional path-based comparison provides nuanced assessment in ambiguous situations. See Appendix A.3 for detailed examples.

6.4 Final Notes

Our metric's bi-directionality explains some disagreements with BLEU and LE. Unlike these unidirectional metrics, we evaluate both directions and select the minimum similarity. This increases confidence in high scores, which require strong matches in both directions. Medium Sim scores are less interpretable, often reflecting asymmetries such as partial coverage or weak predicate matches. High Sim indicates consistent bidirectional agreement, while low Sim reflects structural differences without indicating their origin. Smaller DNF-like trees achieve higher scores when paths match well, but substantial structural differences reduce similarity regardless of tree size, even with nearly identical predicates.

7 Discussion and Future Work

Our systematic analysis reveals several advantages of the Sim metric compared to existing approaches, which we present next. (1) Sim evaluates similarity from a semantic rather than an edit-distance perspective. This allows it to match conceptually related predicates that other metrics miss, offering greater flexibility when evaluating LLM-generated formulae that may use different predicate or argument names. (2) Sim includes both semantic and structural similarity in a single measurement, offering a unified perspective on formal representations. (3) Sim is bidirectional, making it sensitive to performance differences due to path length and tree size variations.

However, our evaluation also highlights important limitations. (1) DNF-like trees remain difficult to process for long formal representations with many AND connected groups under OR, making alignment calculations increasingly expensive. (2) Token embeddings do not guarantee genuine semantic similarities. Embedding pooling can produce high similarity scores for unexpected compounds. When candidates are similar to each other, this creates additional confusion. (3) Current alignment methods are not globally optimal. Our approach now considers only the alignment between the similar predicates, but it does not guarantee an optimal alignment on path-level for best Sim score.

Several promising directions warrant investigation. Domain-specific ontologies could provide more meaningful similarity distinctions, particularly for specialised applications. Alternative embedding approaches that better differentiate irrelevant and contradictory terms may improve matching accuracy. Finally, formulating path matching as a weighted bipartite matching problem and applying algorithms such as the Hungarian algorithm could achieve globally optimal alignments. See Appendix B for detailed discussion.

8 Conclusions

In this paper, we have introduced a novel similarity metric that jointly captures semantic and structural aspects of logical formulae specified in a quantifier-free HOL. Compared with BLEU and LE, our metric better integrated semantic and structural analysis, offering a fuller picture of translation fidelity between natural language and logic which is essential for accurately evaluating how natural language conventions are translated into formal representations within multi-agent systems.

Limitations

In this section, we discuss the limitations and risks of our work.

The dataset and model are designed for English and might not be able to adapt to other languages. Also, performance on different domains needs more thorough analysis.

Although the dataset is claimed to cover different domains, we only work with 1000 pairs. Moreover, due to the features of our HOL, the test set we applied is even smaller, which could raise representative issues.

While we are involving more LLMs in our ongoing experiments, this paper focuses specifically on assessing our proposed metric with BLEU and LE, for which LogicLLama was used and we maintained, to be faithful to that experiment.

Another issue is that the strategy for searching for best matches and the parameters for penalty were decided experimentally with our given dataset. Alternative proposals can be explored in future work.

Concerning computational expenses, the time consumption of our metric seems to be higher than BLEU and LE (but never crosses the 2-5 seconds limit).

The semantic similarity of predicates, as we discuss in the paper, is limited to lexical similarity. Also, since our subset HOL does not support quantifiers, it is difficult to work with complete truth tables, which is also what LE suffers from. Lastly, due to the computational complexity of our metric, our proposal cannot process long inputs with numerous ANDs in a reasonable amount of time.

The generated output's quality is merely decided by the similarity between **Gen** and **GD** that depends on lexical similarity, and not actually assessing whether both **Gen** and **GD** are representing the same concepts. For instance, a low similarity might happen because of the lexical differences in representation, and not because they are representing different concepts.

References

Sarah T. Bachinger, Leila Feddoul, Marianne Jana Mauch, and Birgitta König-Ries. 2024. Extracting legal norm analysis categories from german law texts with large language models. In *Proceedings of the 25th Annual International Conference on Digital Government Research*, dg.o '24, page 481–493, New York, NY, USA. Association for Computing Machinery.

Tina Balke, Célia da Costa Pereira, Frank Dignum, Emiliano Lorini, Antonino Rotolo, Wamberto Vasconcelos, and Serena Villata. 2013. Norms in MAS: Definitions and Related Concepts. In Giulia Andrighetto, Guido Governatori, Pablo Noriega, and Leendert W. N. van der Torre, editors, *Normative Multi-Agent Systems*, volume 4 of *Dagstuhl Follow-Ups*, pages 1–31. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany.

Hossein Rajaby Faghihi, Aliakbar Nafar, Andrzej Uszok, Hamid Karimian, and Parisa Kordjamshidi. 2024. Prompt2demodel: Declarative neuro-symbolic modeling with natural language. In *Neural-Symbolic Learning and Reasoning*, pages 315–327, Cham. Springer Nature Switzerland.

Gabriela Ferraro, Ho-Pun Lam, Silvano Colombo Tosatto, Francesco Olivieri, Mohammad Badiul Islam, Nick van Beest, and Guido Governatori. 2020. Automatic extraction of legal norms: Evaluation of natural language processing tools. In *New Frontiers in Artificial Intelligence*, pages 64–81, Cham. Springer International Publishing.

Yi Fung, Tuhin Chakrabarty, Hao Guo, Owen Rambow, Smaranda Muresan, and Heng Ji. 2023. NORMSAGE: Multi-lingual multi-cultural norm discovery from conversations on-the-fly. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15217–15230, Singapore. Association for Computational Linguistics.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, Lucy Sun, Alexander Wardle-Solano, Hannah Szabó, Ekaterina Zubova, Matthew Burtell, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, and 16 others. 2024. FOLIO: Natural language reasoning with first-order logic. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22017–22031, Miami, Florida, USA. Association for Computational Linguistics.

Amanul Haque and Munindar P. Singh. 2024. Extracting norms from contracts via chatgpt: Opportunities and challenges. *Preprint*, arXiv:2404.02269.

Elias Horner, Cristinel Mateis, Guido Governatori, and Agata Ciabattoni. 2025. From legal texts to defeasible deontic logic via llms: A study in automated semantic analysis. *Preprint*, arXiv:2506.08899.

Samyar Janatian, Hannes Westermann, Jinzhe Tan, Jaromir Savelka, and Karim Benyekhlef. 2023. From text to structure: Using large language models to support the development of legal expert systems. In *Legal Knowledge and Information Systems*, pages 167–176. IOS Press, Maastricht, the Netherlands.

Rushang Karia, Daniel Bramblett, Daksh Dobhal, and Siddharth Srivastava. 2025. Autonomous evaluation of llms for truth maintenance and reasoning tasks. *Preprint*, arXiv:2410.08437.

Youngwoo Kim, Himanshu Beniwal, Steven L. Johnson, and Thomas Hartvigsen. 2025. Decoding the rule book: Extracting hidden moderation criteria from reddit communities. *Preprint*, arXiv:2509.02926.

Chin-Yew Lin and Franz Josef Och. 2004. ORANGE: a method for evaluating automatic evaluation metrics for machine translation. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 501–507, Geneva, Switzerland. COLING.

Junnan Liu. 2025. Few-shot natural language to first-order logic translation via code generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 10939–10960, Albuquerque, New Mexico. Association for Computational Linguistics.

Theo Olausson, Alex Gu, Ben Lipkin, Cedegao Zhang, Armando Solar-Lezama, Joshua Tenenbaum, and Roger Levy. 2023. LINC: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5153–5176, Singapore. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Rajkumar Pujari and Dan Goldwasser. 2025. LLM-human pipeline for cultural grounding of conversations. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1029–1048, Albuquerque, New Mexico. Association for Computational Linguistics.

Shilin Qu, Weiqing Wang, Xin Zhou, Haolan Zhan, Zhuang Li, Lizhen Qu, Linhao Luo, Yuan-Fang Li, and Gholamreza Haffari. 2024. Scalable frame-based construction of sociocultural normbases for socially-aware dialogues. *Preprint*, arXiv:2410.03049.

Tarun Raheja, Raunak Sinha, Advit Deepak, Will Healy, Jayanth Srinivasa, Myungjin Lee, and Ramana Kompella. 2024. Enhancing large language models through transforming reasoning problems into classification tasks. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 6007–6016, Torino, Italia. ELRA and ICCL.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Hyun Ryu, Gyeongman Kim, Hyemin S. Lee, and Eunho Yang. 2025. Divide and translate: Compositional first-order logic translation and verification for complex logical reasoning. *Preprint*, arXiv:2410.08047.

Ramya Keerthy Thatikonda, Jiuzhou Han, Wray Buntine, and Ehsan Shareghi. 2024. Strategies for improving nl-to-fol translation with llms: Data generation, incremental fine-tuning, and verification. *Preprint*, arXiv:2409.16461.

Jidong Tian, Yitian Li, Wenqing Chen, Liqiang Xiao, Hao He, and Yaohui Jin. 2021. Diagnosing the first-order logical reasoning ability through LogicNLI. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3738–3747, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiaoqiang Wang, Lingfei Wu, Tengfei Ma, and Bang Liu. 2024. FAC^{2}E: Better understanding large language model capabilities by dissociating language and cognition. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 13228–13243, Miami, Florida, USA. Association for Computational Linguistics.

Fangzhi Xu, Zhiyong Wu, Qiushi Sun, Siyu Ren, Fei Yuan, Shuai Yuan, Qika Lin, Yu Qiao, and Jun Liu. 2024. Symbol-LLM: Towards foundational symbol-centric interface for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13091–13116, Bangkok, Thailand. Association for Computational Linguistics.

Yuan Yang, Siheng Xiong, Ali Payani, Ehsan Shareghi, and Faramarz Fekri. 2024. Harnessing the power of large language models for natural language to first-order logic translation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6942–6959, Bangkok, Thailand. Association for Computational Linguistics.

May Myo Zin, Ken Satoh, and Georg Borges. 2024. Leveraging llm for identification and extraction of normative statements. In *Legal Knowledge and Information Systems*, pages 215–225. IOS Press, Brno, Czech Republic.

A Detailed Disagreement Case Analysis

A.1 Disagreements With a High Sim

Strong Disagreement cases with a high Sim are rare (0.9% in total). Disagreements usually occur when BLEU fails to capture semantic similarity

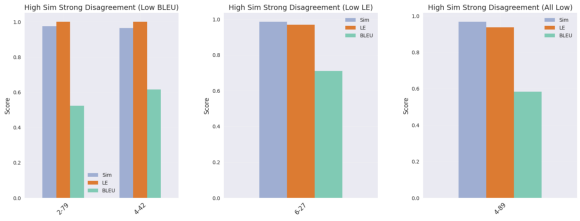

Figure 4: Cases of disagreement with a high Sim.

or when Sim's parser produces different structures than LE's parser (explained in Section 2), as we illustrate with the following selected cases.

High Sim Strong Disagreement (Low BLEU): BLEU assigns low scores to semantically similar cases when predicates are not identical, even when they refer to similar concepts. Since BLEU relies solely on n-gram token overlap, even minor lexical variations, such as missing morphological suffixes (e.g. 's' suffix versus no suffix, case 2-79 of Figure 4) or truncated predicate names (e.g. `CriticallyAcclaimedFilm` versus `CriticallyAcclaimed`, case 4-42 of Figure 4) cause significant drops in BLEU's scrore.

High Sim Strong Disagreement (Low LE): LE is affected by structure differences. In this selected case (case 6-27 of Figure 4), the difference involves only an extra predicate connected by OR (note that **GD** represents the gold standard and **Gen** the generated formula):

$$GD: ((Fruit(x) \wedge Fruit(y) \wedge HigherVitaminC Content(x,y)) \rightarrow BetterPreventScurvy(x,y))$$
$$Gen: ((Fruit(x) \wedge Fruit(y) \wedge HigherVitaminC Content(x) \wedge LowerVitaminCContent(y)) \rightarrow BetterAtPreventingScurvy(x,y))$$

The DNF-like trees have the same size. The only different paths are `[not,lowervitaminccontent,var,y]` from the generated tree (or `LowerVitaminCContent(y)` in the original expression) and `[not, highervitaminccontent,var,y]` from the gold standard tree (or `HigherVitaminCContent(x,y)`). These paths match in both directions. This is very rare, but it happens because the path with the extra y argument in the **GD** and the path with the extra predicate `LowerVitanCContent` in **Gen** end up matching with each other. Thus, LE in this case does capture the structure difference, while Sim fails.

High Sim Strong Disagreement (All Low): In case 4-89 of Figure 4, besides the truncated predicate names affecting BLEU (`PlayedOnField` vs `OnField`), LE is also affected by different parsing strategies. Consider these expressions:

$$GD: ((PlayedWithBall(x) \wedge PlayedOnField(x)) \vee (PlayedWithRacket(x) \wedge PlayedOnCourt(x)))$$
$$Gen: (PlayedWithBall(x) \wedge PlayedOnField(x) \vee PlayedWithRacket(x) \wedge PlayedOnCourt(x))$$

Sim's parser prioritises OR, treating both expressions identically as `[PlayedWithBall(x)` $\wedge$ `PlayedOnField(x),`$\vee$`, PlayedWithRacket(x)` $\wedge$ `PlayedOnCourt(x)]`. Parentheses around the ANDs do not change this structure. LE's parser treats the **GD** expression identically but parses the **Gen** expression as `[PlayedWithBall(x)` $\wedge$ `PlayedOnField(x),`$\vee$`, PlayedWithRacket(x),`$\wedge$`,PlayedOnCourt(x)]`, leading to less similar truth tables.

A.2 Disagreements with a Low Sim

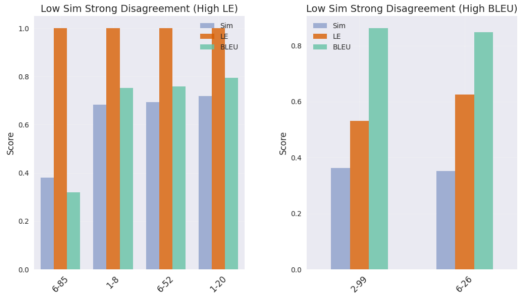

Figure 5: Cases of disagreements with a low Sim.

Disagreements with low Sim cases are slightly more common than those with high Sim cases (6.1%) but still represent a small portion of the total cases. In these cases, a high LE may occur due to LE being insensitive to extra predicate arguments. A high BLEU may occur due to BLEU being insensitive to structural differences behind surface token differences.

Low Sim Strong Disagreement (High LE): Disagreements occur mostly when the LLM generates predicates with arguments embedded in predicate names. For example, in case 6-85 of Figure 5, the LLM generated `ParticipatesInSprints(x)` for the predicate `ParticipatesIn(x,sprints)` of the gold standard, where the argument 'sprints' is moved to the predicate's name in LLM's result.

This often lowers BLEU as well. LE, which binds predicates by finding the closest match based on edit distance, remains largely insensitive to such structural differences. So additional or missing arguments do not strongly affect its score. Sim, however, generates DNF-like tree paths for each predicate argument. Extra arguments introduce additional paths that results in paths being re-matched more often, which in our proposal implies incurring penalties. Differences in path lengths also incur penalties in our approach, preventing Sim from achieving a high score in such cases. In summary, although these two predicates seems similar for a human reader, they do differ in their logical structure (such as the different number of arguments in this example), which both Sim and BLEU reflect.

Some cases with missing arguments still achieve high BLEU scores because the generated HOL representations differ from the gold standard only by the absence of arguments (`ServesMultiplePurposes(x)` versus `ServesMultiplePurposes(x,floodControl,irrigation,hydroelectricPowerGeneration)`), while the remaining structure is identical. In such cases, and again, Sim assigns low scores because the gold standard tree contains three additional paths. For a generated tree with only five paths, this represents a large difference in the numbers of paths of DNF-like trees. This example refer to case 6-52 of Figure 5. If we compare the Sim scores of both cases 6-85 and 6-52, the scores in 6-85 are lower due to the fact that the expression there included a larger number of predicates with similar issues.

Our analysis of the results show that sometimes the Sim approach can lead to mismatches even when nodes are similar. For example, in case 1-8 of Figure 5, we would expect `Romance` and `hasRomance` to have a relatively high score, and similarly with `Comedy` and `hasComedy`. But that is not what happens. Instead, we end up with `Romance`, `Comedy`, `hasRomance`, and `hasComedy` all matching `Movie`. According to cosine similarity, `Movie` embeddings are closer to `Romance` and `Comedy` than `hasRomance` and `hasComedy`. Compound embeddings (like `hasRomance`) are built by pooling multiple subtokens, so they don't align precisely with single-token embeddings (`Romance`). This leads to lower cosine similarity scores and occasional mismatches during path alignment.

Variables in Sim can also result in lower scores. This is because a variable x gets translated into $var(x)$, which results in differences in path lengths, and could also impact the matching of paths. This is the case of 1-20 of Figure 5, where the path `[not,angles,var,x]` gets matched with `[not,polygon,var,x]` instead of `[not,hasangles,var,x]`, preventing optimal matching.

Low Sim Strong Disagreement (High BLEU): High BLEU scores in these cases usually result from HOL strings with few token-level differences, contributing to high n-gram overlap ratios. However, these differences create large logical structure differences that Sim successfully captures.

A common token-level difference involves alternative implication positions, with predicates on different sides of the implication (as in case 2-99 of Figure 5). All predicates remain identical:

$$
\begin{aligned}
&GD: \\
&((FashionChoice(x) \land Trendy(x) \land \\
&Comfortable(x)) \rightarrow (Colorful(x) \lor \\
&InterestingPatterns(x)) \land Stylish(x)) \\
&Gen: \\
&((FashionChoice(x) \land Trendy(x) \land \\
&Comfortable(x) \ \land \ (Colorful(x) \lor \\
&InterestingPatterns(x))) \rightarrow Stylish(x))
\end{aligned}
$$

Since predicates, operators, parentheses, and commas mostly overlap with only few misplacements, BLEU scores remain high.

Another common case involves missing parentheses on the right side of implications, leading to different parsing results. Unlike previous cases where our parser treats OR differently than BLEU and LE parsers, both parsers handle implications identically and correctly: implications only affect predicates or predicate groups directly connected within parentheses. When multiple predicates appear on an implication's right side without encompassing parentheses, only the closest predicate to the implication is affected. For example, in case 6-26 of Figure 5, the LLM makes a mistake by forgetting the parenthesis on the right hand side:

$$
\begin{aligned}
GD: &(Animal(x) \land Reptile(x)) \\
&\rightarrow (HasScales(x) \land LaysEggs(x)) \\
Gen: &(Animal(x) \land Reptile(x)) \\
&\rightarrow HasScales(x) \land LaysEggs(x)
\end{aligned}
$$

Gold standard parsing yields `[Animal(x) ∧ Reptile(x), → , HasScales(x) ∧ LaysEggs(x)]`, while generated parsing produces `[Animal(x) ∧ Reptile(x), → ,HasScales(x), ∧ ,LaysEggs(x)]`, where only `HasScales(x)` connects to the implication. This

creates large logical structure differences. The only score that doesn't get affected here, when it should, is the BLEU score, because missing only a pair of parenthesis does not prevent the rest of the tokens in **Gen** from building n-grams identical to the ones in **GD**.

Low Sim Mild Disagreement (All Medium): This uncommon category (only 1.2% of all cases) typically involves missing predicates and difference in logical structures.

A.3 Disagreements with a medium Sim

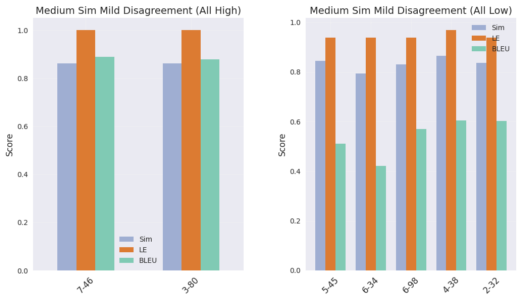

Figure 6: Cases of Disagreement with a medium Sim.

Although a medium Sim mainly results from ambiguity between BLEU and LE, there are some cases where BLEU and LE strongly agree with each other (12.2% in total). These instances primarily reflect the behaviour of Sim's calculation method rather than genuine metric disagreement.

Medium Sim Mild Disagreement (All High): These cases mainly involve mismatches among semantically similar nodes, as mentioned in Low Sim Strong Disagreement (High LE) cases. To provide a couple of examples, we first note that in case 3-80 of Figure 6, structures remain basically identical with more overlapping predicates in both gold standard and generated HOL strings, leading to higher LE and BLEU scores. Sim, however, gets a lower score because not all node matches are semantically appropriate, such as `Trails` being aligned with `Mountain`. On the other hand, in case 7-46 of Figure 6, Sim matches nodes with opposite meanings: `ContainsAlcohol` and `NonAlcoholic`. This occurs because predicate embeddings are pooled from their subtokens, meaning that the resulting vectors may appear close in embedding space even when the underlying meanings differ. Cosine similarity further contributes to this effect: while it can detect opposites to some degree, it does not produce strongly negative similarity for negated or conceptually reversed terms.

Medium Sim Mild Disagreement (All Low): In what follows we present a few more reasons on why Sim may be higher than others. In case 5-45 of Figure 6, `StudiesAndAssessesHumanBehaviorAndMentalProcesses(x)` splits into `StudiesHumanBehaviorAndMentalProcesses(x)` $\wedge$ `AssessesHumanBehaviorAndMentalProcesses(x)`. In such cases, one of the split predicates typically matches the original with high similarity, while the other receives a penalty. However, because Sim computes similarity as the average across all matching paths, this splitting has limited impact on the overall score when the remaining paths are identical or highly similar.

The reverse pattern that multiple predicates connected by operators merge into a single predicate in **Gen** exhibits similar behaviour. For instance, `AppropriateForWinter(x)` $\vee$ `AppropriateForCoolWeather(x)` merged as `AppropriateForWinterOrCoolWeather(x)` (case 6-98 of Figure 6). Beyond these common cases, other merging patterns exist, such as `Atom(y)` $\wedge$ `Atom(z)` merged into `Atoms(y)`, as in case 6-34 of Figure 6. Note that when when $\neg$ precedes a predicate, that predicate may not match the merged predicate. For instance, when `WorksOvertime(x)` $\wedge$ $\neg$`Paid(x)` from **GD** merged into `WorksOvertimeWithoutPay(x)` in **Gen**, $\neg$`Paid(x)` is converted into `[paid,var,x]` in DNF-like tree, which does not match to `[not,worksovertimewithoutpay,var,x]` (case 2-32 of Figure 6).

Similar behaviour occurs when there are extra predicates in longer HOLs (as in case 4-38 of Figure 6):

$$
\begin{aligned}
&GD: \\
&(Prism(x) \rightarrow (DispersesLightIntoColors(x) \\
&\wedge\ RefractsDifferentWavelengths(x) \wedge \\
&VaryingAngles(x)\ \wedge CreatesSpectrum(x))) \\
&Gen: \\
&(Prism(x) \rightarrow \\
&(DispersesLightIntoConstituentColors(x) \wedge \\
&RefractsDifferentWavelengthsOfLight(x) \\
&\wedge\ CreatesSpectrum(x)))
\end{aligned}
$$

Here, an extra pedicate (`VaryingAngles(x)`) exists in **GD**, but other predicates remain almost identical, creating only one additional `[and41,varyingangles,var,x]`. This path can still match to another path with same length and starting with AND. Even though the path similarity

is penalised, it does not substantially affect the Sim score.

However, in the above cases, LE has to use dummies to ensure same length of formulae, which means there is always one predicate from one side that cannot be bound in the CFG tree for truth table generation. Meanwhile, the extra predicate will bring several tokens that do not exist in one side (often tokens from other predicates are not identical), creating many n-grams that are not in **Gen**. These leads to both scores being low.

B Proposed Solutions and Future Research Directions

First, since the matching is semantic-based and embeddings can lead to confusion, we believe that supplementing the system with knowledge representation approaches such as ontologies would be helpful, especially for domain-specific cases. With an additional knowledge base, different concepts would have more distinct and meaningful similarities.

Second, changes to the embedding calculation method could also be beneficial. Currently, cosine similarity is normalised, meaning that nodes that are irrelevant to each other yield a similarity of 0.5. Treating irrelevant and contradictory nodes as having 0 similarity might be better practice when most nodes have similarities higher than 0.

Finally, the final similarity is the average similarity across all paths, where each path's similarity is itself an average of node similarities. When the final similarity is low, either there are no high-similarity matches to the paths, or high-similarity matches have been used by other paths and penalised. Therefore, low similarity can result from either generation quality or matching behaviour. To exclude unexpected matching behaviours, we need to review how predicates are matched. Theoretically, if the matches are accurate, the ratio of high-performance matches among all paths should approximate the final similarity score. When paths are not ordered based on their matching behaviour (i.e., the overall similarity for each predicate group), a path may be used for non-meaningful matches before being used for meaningful ones, leading to similarity scores that are not penalised or are under-penalised. In the rearranged paths, predicates with high-performance matches receive higher priority. However, this still does not guarantee a global optimum. In fact, matching between paths can be formulated as a weighted bipartite matching problem based on path similarity. We plan to apply algorithms such as the Hungarian algorithm for this matching task.