SR-LLM: Rethinking the Structured Representation in Large Language Model

Anonymous ACL submission

Abstract

Structured representations, e.g., Abstract Meaning Representation (AMR), have long been pivotal in computational linguistics. However, their role in the era of Large Language Models (LLMs) remains ambiguous. Initial attempts to integrate structured representation into LLMs via zero-shot setting yielded inferior performance. We hypothesize that such a decline stems from the structure information being passed into LLMs in a code format unfamiliar to LLMs' training corpora. To this end, we propose SR-LLM, an innovative framework with two settings to explore the optimal way of integrating structured representation with LLMs from both training-free and training-dependent perspectives. The former embeds the natural language description of structural information into the prompts of LLMs, while the latter leverages task-specific structured datasets to fine-tune LLMs. Across ten wide-recognized datasets, our training-free and training-dependent approaches yield about 3% to 10% performance increase on average. We further extend our approach to alternative structure representations other than AMR, such as Parse Syntax Trees and First-Order Logic. To the best of our knowledge, we firstly show that structure representation can boost the performance of LLMs. We hope our work sheds light to encourage future research to enhance the reasoning and interoperability of LLMs. We will make our code publicly available upon acceptance of this work.

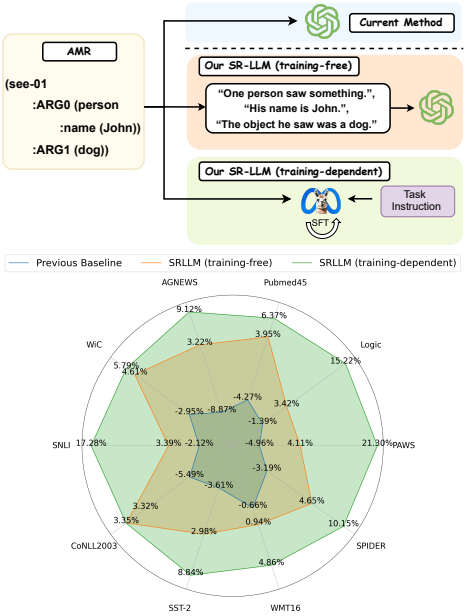


Figure 1: Our SR-LLM framework in Llama3.1-8b-Instruct outperforms previous baseline. **(Top)**Comparing to the existing methods where such structured representation is passed to LLMs in the prompt as a code format, we propose two settings, a training-free one where we convert AMR to natural language, and a training-dependent one where we fine-tune LLMs with AMR and task instructions. **(Bottom)**We show that our method outperforms previous methods by a large margin over 10 different tasks, on average of 3-10%.

1 Introduction

Natural language, while crucial for human communication and knowledge accumulation (Pinker and Bloom, 1990; Khurana et al., 2023), faces limitations due to its logical vagueness and semantic ambiguity (Raskin et al., 2013; Yadav et al., 2021). This has long been a challenge in Natural Language Processing(NLP) (Chowdhary and Chowdhary, 2020; Nadkarni et al., 2011; Ramírez, 2024).

As an efficient method of representing knowledge, structured representation(SR) mitigate these limitations of natural language and have long been a cornerstone in the field of NLP (Chowdhary and Chowdhary, 2020; Nadkarni et al., 2011; Ramírez, 2024), playing a crucial role in advancing both its theoretical foundations (Manning, 1999) and practical applications (Collobert et al., 2011). SR such as

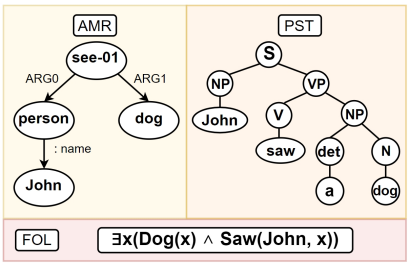


Figure 2: The AMR, PST, and FOL of the sentence "John saw a dog".

Abstract Meaning Representation (AMR) (Damonte et al., 2016; Knight et al., 2021; Ramírez, 2024), Parse Syntax Trees (PST) (Sachan et al., 2020), and First-Order Logic (FOL) (Barwise, 1977) provide sophisticated frameworks for text analysis, capable of capturing complex semantic relationships and facilitating nuanced understanding of linguistic content (Banarescu et al., 2013; Wang et al., 2015). An example of AMR, PST, and FOL is depicted in Figure 2.

However, with the advent of LLMs, uncertainty persists regarding the optimal utilization of SR. Recent studies attempting to directly integrate AMR into prompts have yielded results indicating that this approach often proves detrimental rather than beneficial to performance (Jin et al., 2024). Nonetheless, we hypothesize that the observed inefficacy stems from the LLMs' inherent inability to comprehend and effectively utilize structured representations. The direct incorporation of these complex linguistic structures into the reasoning process appears to introduce unintended cognitive artifacts, potentially obfuscating rather than elucidating the underlying semantic and logical relationships. Moreover, we posit that models with lower performance capabilities are more susceptible to these adverse effects, exacerbating the issue for less sophisticated LLMs. This misalignment between the structured input and the LLMs' processing mechanisms likely results in detrimental side effects, compromising the models' reasoning capabilities rather than enhancing them.

To address these challenges and effectively integrated structured representations into LLMs, we propose a novel framework called SR-LLM, which encompasses two configurations: training-free and training-dependent. In the training-free setting, we employ rule-based methods to convert SR into natural language description(SR-NLD), then incorporating it into prompts to assist LLMs in reasoning. This approach enhances the comprehensibility of structured information for LLMs, augmenting the input with rich semantic, syntactic, and logical features. By preserving the essence of structured representations in a more accessible format, SR-NLD facilitates a more nuanced understanding of text, reduce ambiguity, and promotes structured thinking. Furthermore, this method seamlessly integrates with LLMs' pre-existing knowledge, circumventing the need for additional training on specific structured representations. Consequently, SR-NLD serve as a bridge between complex structured data and LLMs' natural language processing capabilities, ultimately fostering more accurate and sophisticated reasoning across various NLP tasks.

The training-dependent setting involves supervised fine-tuning (SFT) of LLMs on task-specific SR datasets (termed Gen-SR) to establish connections between structured information and tasks. During the process of SFT, the LLM is exposed to structured data repeatedly, enabling it to develop a more robust internal representation of the underlying structures. This deep integration allows the model to leverage structured knowledge during inference, improving its performance across a wide range of NLP tasks. In essence, the training-dependent approach helps bridge the gap between LLMs' general language capabilities and the precise, structured understanding required for complex reasoning tasks.

We conducted extensive evaluations across a diverse array of NLP tasks, encompassing 10 distinct datasets that cover a wide spectrum of language processing challenges. These tasks include paraphrase detection (Mihalcea et al., 2006; Dolan and Brockett, 2005), recognizing textual entailment (Dagan et al., 2005; Bowman et al., 2015), machine translation (Bahdanau, 2014; Johnson et al., 2017), etc. This comprehensive selection ensures a thorough assessment of our approach across various aspects of natural language understanding and generation. The results of our experiments demonstrate a clear contrast between existing methods and our proposed approaches. While previous techniques led to a 3% decline in average performance, our training-free method

showed a positive shift, improving performance by 3% on average. Even more impressive, our training-dependent approach exhibited a substantial 10% overall performance boost across tasks, with a particularly striking 21.3% improvement on the PAWS dataset. These outcomes provide compelling evidence that LLMs can significant enhance their capability when presented with structured information in an accessible and appropriate manner.

Our contributions are as follows:

- We propose SR-LLM, which is a novel approach to integrate structured representations (SR) into large language models (LLMs), exploring both training-free and training-dependent settings.
- We demonstrate the effectiveness of our approach across 10 diverse NLP tasks, showing an average 3% improvement in the training-free setting and a 10% improvement in the training-dependent setting, with a notable 21.3% boost on the PAWS dataset.
- We provide insights into how different types of structured information(AMR, PST, FOL) impact LLMs performance across various tasks.
- To the best of our knowledge, we are the first to show that combining such structured representations can actually boost the performance of LLMs, opening new avenues for enhancing LLM reasoning and interoperability.

2 Problem Definition

This research endeavors to investigate the potential synergies between SR and LLMs, with the ultimate goal of ascertaining how their seamless integration can augment the efficacy and proficiency of LLMs in a wide array of NLP tasks. Given a natural language input sequence represented as $X = (x_1, x_2, \ldots, x_n)$, where $x_i \in V$ denotes a token drown from the vocabulary V. The primary objective is to establish a mapping from this intricate input to a corresponding output sequence $Y = (y_1, y_2, \ldots, y_m)$, where each y_i represents elements drown from either the target vocabulary or a structured semantic output space. This mapping is executed by the model f according to the following formulation:

$$Y = f(X) \tag{1}$$

This sophisticated mapping process encompasses a hierarchical cascade of linguistic processing layers, incorporating lexical analysis, syntactic parsing, semantic interpretation, and pragmatic inference, each contributing to the multifaceted transformation of input to output.

Complementing the natural language input, we introduce the concept of structured representation Z. This representation can manifest in various forms, including but not limited to AMR, PST, or FOL, each capturing distinct facets of semantic, syntactic, or logical information respectively. To facilitate this transformation, we define a function g that maps the natural language input X into its corresponding structured representation Z:

$$Z = g(X) \tag{2}$$

To quantify the benefits conferred by the incorporation of SR, we introduce an "SR enhancement" function, denoted as Δ, which measures the performance differential between two scenarios: one utilizing solely the natural language input X, and another leveraging both the natural language input and its corresponding structured representation X, Z. The enhancement function is formally defined as follows:

$$\Delta = P(f(X, Z)) - P(f(X)) \tag{3}$$

where $P(\cdot)$ denotes a performance evaluation metric such as accuracy or F1 score. The objective of this investigation is to identify an optimal model f^* that can effectively utilize both the natural language input and its corresponding structured representations to maximize the performance gain Δ:

$$f^* = \arg\max_f \Delta \tag{4}$$

3 Method

This chapter introduces the SR-LLM framework, a novel paradigm designed to investigate the efficacious integration of SR into LLMs. The primary objective is to augment LLM performance across a diverse spectrum of NLP tasks. The SR-LLM framework encompasses

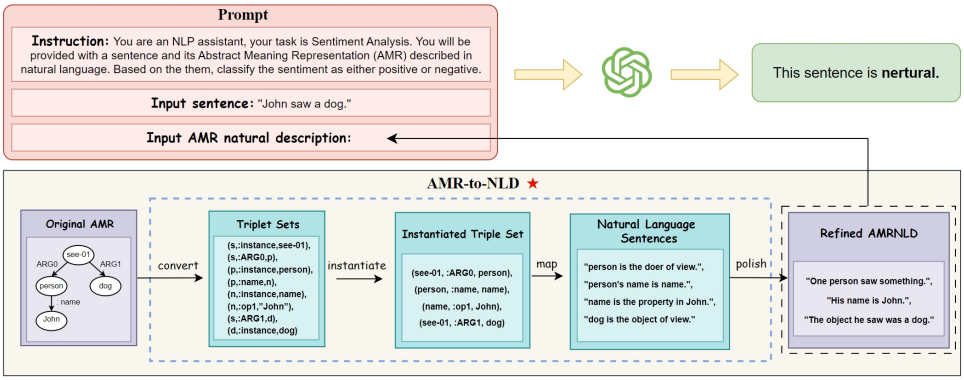


Figure 3: **The whole process of SR-LLM in training-free setting**. Initially, a task-specific prompt consists of an instruction, input sentence, and input SR structure (AMR is used here). Subsequently, the original AMR undergoes transformation via the **AMR-to-NLD** module, which employs predefined rules to map the AMR into an easily interpretable natural language description. This description is then subjected to refinement by a language model, ensuring fluency and coherence, resulting in **AMR-NLD**. Finally, the **AMR-NLD** is seamlessly integrated into the input, which is then fed into the LLM to generate the ultimate response.

two configurations: training-free and training-dependent. These configurations are designed to amalgamate various types of SR through differentiated methodologies, thereby enhancing the LLMs' capability to comprehend and exploit structured information. The ultimate goal is to maximize the structured information improvement, quantified by Δ.

3.1 Preliminary

The structure representations of AMR, PST, and FOL each have their unique advantages and applications in semantic abstraction, syntactic parsing, and logical reasoning, respectively. AMR is rooted, labeled graph structures designed to abstract away from syntactic nuances, providing concise and semantically rich representations (Banarescu et al., 2013). PST is a hierarchical tree structure grounded in Chomsky's theory of generative grammar, commonly used to represent the syntactic organization of sentences, emphasizing the dependency between words (Chomsky, 2014). FOL is a symbolic logic system that articulate objects, their relations, and properties, making them essential tools in formal logic and logical reasoning (Enderton, 2001; Barwise, 1977).

3.2 SR-LLM Training-Free

Prior approaches, exemplified by AMRCOT (Jin et al., 2024), have attempted to explicitly incorporate AMR into Chain-of-Thought (COT) prompts, as illustrated in Figure 4, have shown that this explicit approach fails to yield

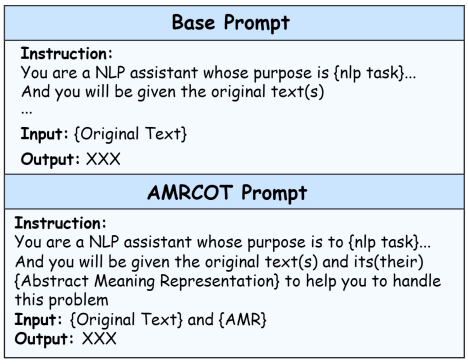


Figure 4: **Base prompt and AMRCOT prompt. (Top)** This is the original task prompt, with only the raw text as input, serving as the standards for performance. By comparing the results, we can observe whether different methods introducing SR can lead to performance improvements. **(Bottom)** This is the AMRCOT prompt method proposed by Jin et al. (2024), serving as a baseline. By comparing with it, we aim to evaluate if our proposed method shows any advancements.

performance enhancement. We hypothesize that one factor contributing to this ineffectiveness stems from the inherent difficulty LLMs face in adequately comprehending and processing abstract structures such as AMR. In view of the aforementioned challenge, as illustrated in Figure 3, we propose SR-LLM Training-Free, where the original structured representation Z is transformed into natural language descriptions termed **SR-NLD**, where SR can be instantiated with specific structured representations such as AMR, PST, and FOL. The transformation is achieved through a rule-based algorithm, and the resulting natural language

description is subsequently integrated with the original input X. This enables LLMs to directly utilize the rich information embedded within the SR during inference. A pivotal advantage of this methodology lies in its training-free nature, as it does not require any additional fine-tuning or retraining of the LLM. Consequently, this technique offers remarkable flexibility, enabling rapid adaption to a diverse array of NLP tasks.

We refer to this entire transformation process as **SR-to-NLD**(**S**tructured **R**epresentation **to** **N**atural **L**anguage **D**escription). Specifically, the structured representations are mapped through predefined transformation rules, converting abstract symbols into easily interpretable natural language expressions. These generated natural language descriptions are then refined by a language model to ensure fluency and coherence.

Next, we shall elucidate the SR-to-NLD process, employing AMR-NLD as our quintessential exemplar, which shown in the Algorithm 1. The process first converts the AMR graph into triplets, then replaces the identifiers with actual concepts. Next, the triplets are mapped into natural language descriptions using predefined rules, and finally, the descriptions are refined by a language model to produce coherent AMR natural language descriptions (AMR-NLD). This detailed analysis forms the core of our discussion, outlining each step of the conversion process. The transformation methods for other SRs are elaborated in the Appendix A.2 for completeness.

3.3 SR-LLM Training-Dependent

In addition to making SRs more interpretable for LLMs, we also believe that establishing connections between tasks and structured information presents a potential opportunity. As shown in the Figure 5, in SR-LLM Training-Dependent, we constructed a task-specific hybrid dataset, named Gen-SR, where SR can be replaced by specific representations such as AMR, PST, and FOL.

The entire hybrid dataset is composed of two equal parts: one consists of task-specific instruction pairs based on the original text, while the other adds structured representations in the instruction pairs based on the former. The former we mark as G(text) and the other we mark as G(SR). The complete example

Algorithm 1 AMR-to-NLD Transformation

Input: AMR graph $G = (V, E)$, nodes collection V, edges collection E, Penman library $\mathcal{P}$, language model θ
Output: Refined natural language descriptions S_{refined}
***Phase 0**: Convert AMR to Triplets*
Convert AMR graph G into triplets $T = \{(c_1, r, c_2) \mid c_1, c_2 \in V, r \in E\}$ using the Penman library: $T = \mathcal{P}(G)$
***Phase 1**: Identifier Instantiation*
for each triplet $(c_1, r, c_2) \in T$ **do**
    if $r =$ `:instance` **then**
        Replace identifiers c_1, c_2 with their corresponding concepts or instances
    end if
end for
***Phase 2**: Mapping to Natural Language*
Convert triplets into natural language descriptions using a predefined dictionary: $M : T' \rightarrow S$
***Phase 3**: Refinement*
Refine the generated descriptions S using language model: $S_{\text{refined}} = \theta(S)$
return S_{refined}

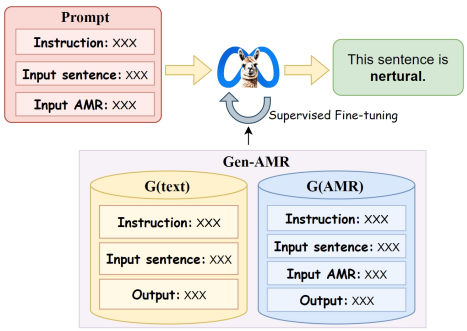
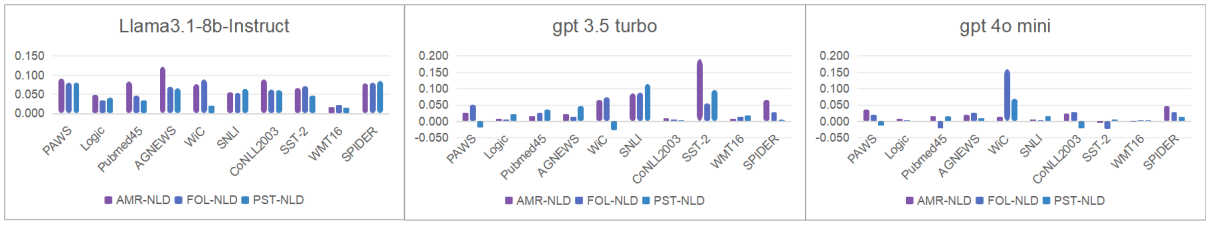
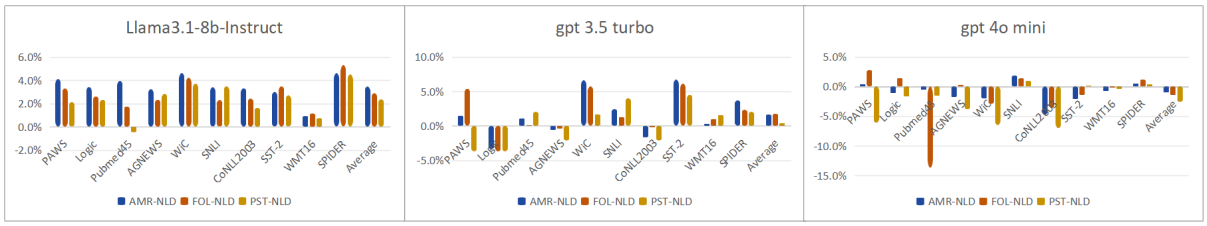


Figure 5: **The whole process of SR-LLM in training-dependent setting**. Taking AMR as an example, a dataset called **Gen-AMR**, created by combining inputs consisting of sentences and their corresponding AMR structures, is utilized for the SFT of LLM to enhance the reasoning capability.



(a) Difference between SR-LLM(training-free) and Previous Baseline



(b) SR enhancement (Δ) of SR-LLM(training-free)

Figure 6: **Performance of SR-LLM(training-free).**(a) The performance difference is calculated by subtracting the average performance of the Current Method from the average performance of SR-LLM(training-free) across both CoT and One-Shot strategies for the same task. In cases where the performance difference is positive, it indicates that SR-LLM(training-free) outperforms the Current Method. (b) This enhancement is calculated by subtracting the base performance (where the prompt contains only the original text without SR) from the average performance of SR-LLM(training-free) under both CoT and One-Shot strategies for the same task. Similarly, a positive SR enhancement value indicates that SR-LLM(training-free) performs better than the baseline.

of these two are shown in the Appendix D. This mixed approach allows the LLM to not only learn instruction-following for downstream tasks but also to establish more robust connections between tasks and structured representations, making the model achieve more effective improvements compared to learning solely from raw text.

4 Experiments

4.1 Datasets

To ensure comprehensive and diverse experiments, we selected 10 datasets covering various NLP tasks based on Liu et al. (2024), including five tasks from Jin et al. (2024) for result comparability. The dataset composition includes: PAWS for paraphrase detection (Zhang et al., 2019), SNLI for textual entailment recognition (Bowman et al., 2015), WMT16 for translation tasks (Bojar et al., 2016), CoNLL2003 for named entity recognition (Sang and De Meulder, 2003), Logic for logical fallacy detection (Jin et al., 2022), SST-2 for sentiment analysis (Socher et al., 2013), Pubmed45 for event extraction (Garg et al., 2016), WiC for word sense disambiguation (Pilehvar and Camacho-Collados, 2018), SPIDER for Text2SQL code generation (Yu et al., 2018), and AGNEWS for text classification (Zhang et al., 2015). The detailed collection processes and results provided in the Appendix B.1 and Appendix B.2.

4.2 Training-Free Results

Experimental Details. We conducted experiments on the Llama3.1-8b-Instruct, GPT-3.5-turbo, and GPT-4o-mini models, arranged from weak to strong based on their performance levels, employing two prompting strategies: Chain-of-Thought (CoT) (Wei et al., 2022) and One-Shot (Brown, 2020). CoT guides step-by-step reasoning, while One-Shot demonstrates task-solving through specific examples. For detailed prompt, refer to the Appendix E.

Result Analysis. First, as shown in the part (a) of Figure 6, the results demonstrate that SR-LLM(training-free) generally outperforms previous baseline. The best and most stable performance observed is in the smaller Llama3.1-8b-Instruct, where average performance improvement in across different SRs ranges from 3% to 10%. This suggests that for LLMs, converting abstract structured representations into a natural language format, which the model is more familiar with, is an effective strategy to

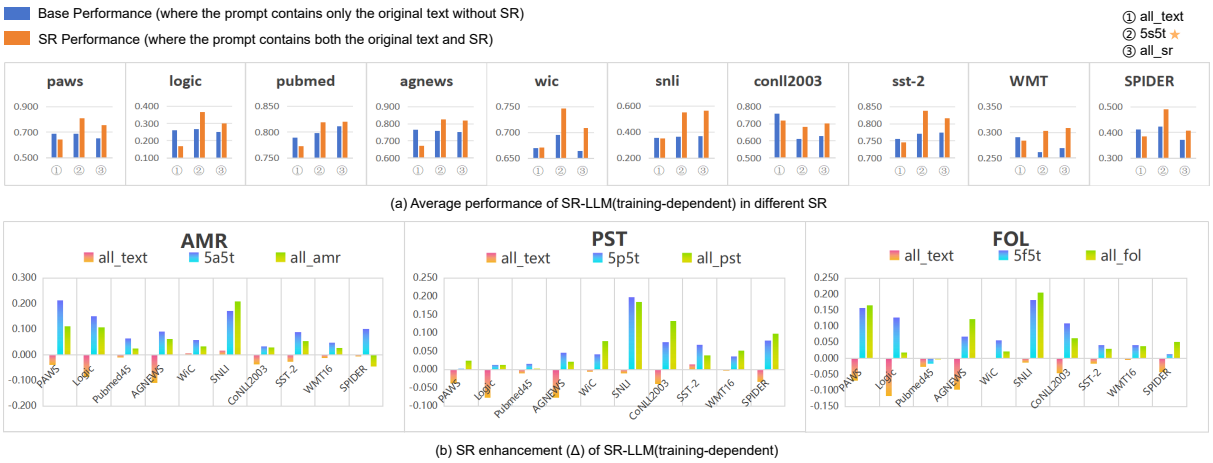


Figure 7: **Performance of SR-LLM(training-dependent). "5s5t"** represents our main model whose training set composed of 50% G(text) and 50% G(SR), while **"all_text"** represents the comparison group whose training set composed entirely of G(text). And **"all_sr"** refers to another comparison group training set composed entirely of G(SR). (a) shows the average performance across the three structured representations (AMR, PST, and FOL). (b) illustrates the SR enhancement in SR-LLM(training-dependent).

enhance the model's ability to interpret and apply structured information.

Next, turning to the part (b) of Figure 6, we observe a gradual decrease in the benefit of structured information as model performance increases. This result further illustrates that weaker models benefit more from structured information as a supplement to the original text, aiding them in downstream reasoning tasks. In contrast, for stronger models, the additional structured information offers limited advantages and may even be less informative than the insights derived directly from the raw text. Moreover, the potential errors and noise introduced by processing structured data can sometimes mislead the reasoning process of the LLM.

4.3 Training-Dependent Results

Experimental Details We conducted experiments using the Llama3.1-8B-Instruct model to evaluate the performance of the training-dependent setting of SR-LLM. Detailed data collection procedures and specific training data configurations are provided in the Appendix B.2. To ensure a more comprehensive comparison, we also performed two sets of comparative experiments: one where all instruction pairs are solely G(text), and the other where all instruction pairs are G(SR).

Result Analysis. From the results in the part (b) of Figure 7, it is evident that when the training data contains a certain proportion of structured representations, the model consistently outperforms those trained solely on text in downstream tasks. Additionally, as shown in the part (b) of Figure 7, SR-LLM(training-dependent) achieves substantial SR enhancement across most tasks, with gains typically exceeding 10%. In some cases, the improvements are even greater than 20%, such as in PAWS trained with AMR and SNLI trained with PST. These findings suggest that when the model establishes strong associations between tasks and structured representations during training, it is able to leverage this information to facilitate more effective reasoning during inference.

Conversely, the comparison shows that when no structured representations are included in the training data, the performance gains are nearly negative. Moreover, when the training data consists entirely of structured representations, the performance is not as strong as when there is a balanced mix of text and structured data. This indicates that a balanced integration of raw text and structured representations is crucial for maximizing the model's reasoning capabilities.

5 Related Work

Structured Representations used for NLP in Pre-LLM Before the advent of LLMs, AMR's graph-based structures were essential in NLP tasks like paraphrase detection and machine translation due to their ability to encode deep semantics and integrate external knowledge (Tohidi and Dadkhah, 2022). For example, (Lim et al., 2020) improved reasoning by combining AMR with ConceptNet, while (Zhang et al., 2021) and (Xu et al., 2022) enhanced information extraction by integrating AMR with external sources. PSTs have been widely used in linguistic analysis, aiding tasks like semantic role labeling and improving predictions through parse tree analysis (Gildea and Palmer, 2002). (Collins, 2003) and (Chiang, 2005) improved parsing and translation accuracy using PSTs. FOLs are key for formal reasoning, providing a structured framework. (Bursztyn et al., 2015) improved query efficiency through FOL reformulations, and (Lalwani et al., 2024) enhanced logical fallacy detection by converting natural language into FOL formulas. Before the rise of large language models (LLMs), natural language mapping to structured representations relied on rule-based methods, which faced semantic limitations(Thompson et al., 1969; Wang et al., 2018).

Structured Representations used for NLP in LLM With the rise of LLM, studies like (Hahn et al., 2022) showed these sequence to sequence model's ability to generalize across formal domains, though challenges like low interpretability and hallucinations persist(De Bellis, 2023). Integrating structured representations into LLMs has improved accuracy and interpretability. (Yao et al., 2024) and (Shi et al., 2024) combined AMR with LLMs for tasks like sentence simplification and Retrieval-Augmented Generation. (Yang et al., 2024) used AMR to strengthen dialogue systems against adversarial inputs. Additionally, (Hahn et al., 2022) and (Kalyanpur et al., 2024) advanced formal specification and logical reasoning in LLMs. However, (Jin et al., 2024) argued that AMR might sometimes hinder performance in certain NLP tasks.

6 Conclusion

SR-LLM demonstrates significant progress in enhancing LLMs' reasoning capabilities through structured representations. Our evaluation across diverse NLP tasks revealed SR's potential in generating novel implicit information. We established a framework for integrating SR into LLMs, from prompt engineering to fine-tuning, providing valuable insights into structured information incorporation. The proposed synthetic structured datasets showed exceptional performance in augmenting LLM capabilities. These advancements led to substantial improvements in both training-free and training-fependent settings, highlighting the effectiveness of integrating semantic, syntactic, and logical features. As we refine SR-LLM, we anticipate further progress towards more interpretable, accurate, and versatile language models with enhanced reasoning capabilities across various NLP applications.

7 limitations

Despite SR-NLD's promising performance in certain tasks, its effectiveness remains inconsistent across different LLMs and lacks broad generalizability. The rule-based conversion method may constrain flexibility. Future research should focus on developing a more robust and adaptive structured representation, exploring task-specific optimizations, and investigating advanced conversion techniques and novel model architectures. Expanding evaluation to diverse language models and datasets will be crucial to enhance the method's consistency, flexibility, and applicability across various NLP domains.

References

Dzmitry Bahdanau. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th linguistic annotation workshop and interoperability with discourse*, pages 178–186.

Jon Barwise. 1977. An introduction to first-order

logic. In *Studies in Logic and the Foundations of Mathematics*, volume 90, pages 5–46. Elsevier.

Ondrej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, et al. 2016. Findings of the 2016 conference on machine translation (wmt16). In *First conference on machine translation*, pages 131–198. Association for Computational Linguistics.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Tom B Brown. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Damian Bursztyn, François Goasdoué, and Ioana Manolescu. 2015. Efficient query answering in dl-lite through fol reformulation. In *28th International Workshop on Description Logics*.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd annual meeting of the association for computational linguistics (acl'05)*, pages 263–270.

Noam Chomsky. 2014. *Aspects of the Theory of Syntax*. 11. MIT press.

KR1442 Chowdhary and KR Chowdhary. 2020. Natural language processing. *Fundamentals of artificial intelligence*, pages 603–649.

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational linguistics*, 29(4):589–637.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12:2493–2537.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine learning challenges workshop*, pages 177–190. Springer.

Marco Damonte, Shay B Cohen, and Giorgio Satta. 2016. An incremental parser for abstract meaning representation. *arXiv preprint arXiv:1608.06111*.

Alessandro De Bellis. 2023. Structuring the unstructured: an llm-guided transition. In *DC@ ISWC*.

Bill Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Third international workshop on paraphrasing (IWP2005)*.

Herbert B Enderton. 2001. *A mathematical introduction to logic*. Elsevier.

Sahil Garg, Aram Galstyan, Ulf Hermjakob, and Daniel Marcu. 2016. Extracting biomolecular interactions using semantic parsing of biomedical text. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Daniel Gildea and Martha Palmer. 2002. The necessity of parsing for predicate argument recognition. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 239–246.

Christopher Hahn, Frederik Schmitt, Julia J Tillman, Niklas Metzger, Julian Siber, and Bernd Finkbeiner. 2022. Formal specifications from natural language. *arXiv preprint arXiv:2206.01962*.

Zhijing Jin, Yuen Chen, Fernando Gonzalez, Jiarui Liu, Jiayi Zhang, Julian Michael, Bernhard Schölkopf, and Mona Diab. 2024. Analyzing the role of semantic representations in the era of large language models. *arXiv preprint arXiv:2405.01502*.

Zhijing Jin, Abhinav Lalwani, Tejas Vaidhya, Xiaoyu Shen, Yiwen Ding, Zhiheng Lyu, Mrinmaya Sachan, Rada Mihalcea, and Bernhard Schoelkopf. 2022. Logical fallacy detection. *arXiv preprint arXiv:2202.13758*.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Aditya Kalyanpur, Kailash Saravanakumar, Victor Barres, Jennifer Chu-Carroll, David Melville, and David Ferrucci. 2024. Llm-arc: Enhancing llms with an automated reasoning critic. *arXiv preprint arXiv:2406.17663*.

Diksha Khurana, Aditya Koli, Kiran Khatter, and Sukhdev Singh. 2023. Natural language processing: state of the art, current trends and challenges. *Multimedia tools and applications*, 82(3):3713–3744.

Kevin Knight, Bianca Badarau, Laura Baranescu, Claire Bonial, Madalina Bardocz, Kira Griffitt, Ulf Hermjakob, Daniel Marcu, Martha Palmer, Tim O'Gorman, et al. 2021. Abstract meaning representation (amr) annotation release 3.0.

Abhinav Lalwani, Lovish Chopra, Christopher Hahn, Caroline Trippel, Zhijing Jin, and Mrinmaya Sachan. 2024. Nl2fol: Translating natural language to first-order logic for logical fallacy detection. *arXiv preprint arXiv:2405.02318*.

Jungwoo Lim, Dongsuk Oh, Yoonna Jang, Kisu Yang, and Heuiseok Lim. 2020. I know what you asked: Graph path learning using amr for commonsense reasoning. *arXiv preprint arXiv:2011.00766*.

Yang Liu, Jiahuan Cao, Chongyu Liu, Kai Ding, and Lianwen Jin. 2024. Datasets for large language models: A comprehensive survey. *arXiv preprint arXiv:2402.18041*.

Christopher D Manning. 1999. *Foundations of statistical natural language processing*. The MIT Press.

Rada Mihalcea, Courtney Corley, Carlo Strapparava, et al. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *Aaai*, volume 6, pages 775–780.

Prakash M Nadkarni, Lucila Ohno-Machado, and Wendy W Chapman. 2011. Natural language processing: an introduction. *Journal of the American Medical Informatics Association*, 18(5):544–551.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2018. Wic: the word-in-context dataset for evaluating context-sensitive meaning representations. *arXiv preprint arXiv:1808.09121*.

Steven Pinker and Paul Bloom. 1990. Natural language and natural selection. *Behavioral and brain sciences*, 13(4):707–727.

José Gabriel Carrasco Ramírez. 2024. Natural language processing advancements: Breaking barriers in human-computer interaction. *Journal of Artificial Intelligence General Science (JAIGS) ISSN: 3006-4023*, 3(1):31–39.

Victor Raskin, Julia M Taylor, and Lauren M Stuart. 2013. Is natural language ever really vague? a computational semantic view. In *2013 IEEE International Conference on Cybernetics (CYBCO)*, pages 74–79. IEEE.

Devendra Singh Sachan, Yuhao Zhang, Peng Qi, and William Hamilton. 2020. Do syntax trees help pre-trained transformers extract information? *arXiv preprint arXiv:2008.09084*.

Erik F Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*.

Kaize Shi, Xueyao Sun, Qing Li, and Guandong Xu. 2024. Compressing long context for enhancing rag with amr-based concept distillation. *arXiv preprint arXiv:2405.03085*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Frederick B Thompson, Peter C Lockemann, B Dostert, and RS Deverill. 1969. Rel: A rapidly extensible language system. In *Proceedings of the 1969 24th national conference*, pages 399–417.

Nasim Tohidi and Chitra Dadkhah. 2022. A short review of abstract meaning representation applications. *Modeling and Simulation in Electrical and Electronics Engineering*, 2(3):1–9.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015. A transition-based algorithm for amr parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 366–375.

Qingxiang Wang, Cezary Kaliszyk, and Josef Urban. 2018. 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*, pages 255–270. Springer.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Runxin Xu, Peiyi Wang, Tianyu Liu, Shuang Zeng, Baobao Chang, and Zhifang Sui. 2022. A two-stream amr-enhanced model for document-level event argument extraction. *arXiv preprint arXiv:2205.00241*.

Apurwa Yadav, Aarshil Patel, and Manan Shah. 2021. A comprehensive review on resolving ambiguities in natural language processing. *AI Open*, 2:85–92.

Bohao Yang, Kun Zhao, Chen Tang, Liang Zhan, and Chenghua Lin. 2024. Structured information matters: Incorporating abstract meaning representation into llms for improved open-domain dialogue evaluation. *arXiv preprint arXiv:2404.01129*.

Peiran Yao, Kostyantyn Guzhva, and Denilson Barbosa. 2024. Semantic graphs for syntactic simplification: A revisit from the age of llm. *arXiv preprint arXiv:2407.04067*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019. Paws: Paraphrase adversaries from word scrambling. *arXiv preprint arXiv:1904.01130*.

Zixuan Zhang, Nikolaus Nova Parulian, Heng Ji, Ahmed S Elsayed, Skatje Myers, and Martha Palmer. 2021. Fine-grained information extraction from biomedical literature based on knowledge-enriched abstract meaning representation. In *Proc. The Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2021)*.

@articlepinker1990natural, title=Natural language and natural selection, author=Pinker, Steven and Bloom, Paul, journal=Behavioral and brain sciences, volume=13, number=4, pages=707–727, year=1990, publisher=Cambridge University Press

A Details of Converting SR to SR-NLD

A.1 Details of Translating AMR Triplet to Natural Sentence

According to the Figure 8, first, the triplet is converted into a sentence based on the relation mapping rules. Then, using the entity dictionary, the entities are replaced with their actual meanings to form the final sentence. Finally, the sentence is input into the LLM for refinement into a complete and coherent sentence, as shown in the Figure 9.

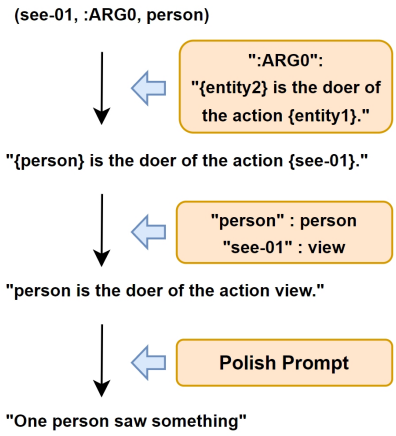


Figure 8: The process of translate entities and relationships into natural language sentences

A.2 Whole Process of Making PST-NLD

Definition of PST. PST is represented as a tree structure $T = (N, E)$. Here N denotes the set of nodes, representing the syntactic components of a sentence (e.g., part-of-speech tags and phrase labels). Node types include S (sentence), NP (noun phrase), VP (verb phrase), etc. E denotes the set of edges, representing dependencies between components. An example of the original PST structure is shown in the Figure 10.

Conversion of PST to a Linear Structure Using Depth-First Search (DFS). Starting from the root node (typically n_0, representing the sentence's syntactic structure, such as S), we traverse the tree in a depth-first search (DFS) manner, converting it into a linear sequence of symbols P.

Mapping PST Identifiers to Natural Language Descriptions. We define a mapping function M to translate each identifier (e.g., S, NP, VBD) and its child nodes into natural language descriptions. The dictionary D, which specifies the natural language interpretation of each identifier, is detailed in the appendix. For each triplet (n, c_1, c_2), where n is a node and c_1, c_2 are its children, we apply the mapping function $M(n) = \text{description}(n)$. The resulting natural language description S is as shown in the Figure 10.

Refinement of Natural Language Descriptions Using a Language Model. To make the descriptions more natural and coherent, the generated descriptions S are refined using the language model $F_{\text{LM}} : S \rightarrow S_{\text{refined}}$. The specific prompt is shown in the prompt (b) of Figure 11.

Make_natural_amr_information

[system_prompt]
Your task is to polish a collection of sub sentences and rewrite them based on the original sentence to make them read more like natural language.
Please output your answer according to the example.
###
Example 1:
input_original_sentence: "In 2014, the site launched iOS and Android applications for product search, product features include live - video - product - reviews with interactive questions and answers - sessions.",
inoput_sub_sentences: "
0 launch-01(set off, introduce, bring up, start, take off) is the first sub-sentence of multi-sentence.
1 site is the doer of launch-01(set off, introduce, bring up, start, take off).
2 and is the object of launch-01(set off, introduce, bring up, start, take off).
3 and's purpose is search-01(search, seek, seaking, looking through/for).
...
26 feature is descriped as product.
",
output sentences: "
0 Launching is the first action in a series of actions.
1 The site is the one that launched.
2 The site launched applications.
3 The purpose of the applications is to enable product search.
...
26 The features are related to products.
###
Example 2:
...

[user_prompt]
input_original_sen: `{sentence}`,
inoput_sub_sen: `{ori_sub_sentences}`,
output sentences:

Figure 9: The prompt of polishing sentence for making AMR-NLD

A.3 Whole Process of Making FOL-NLD

Definition of FOL. FOL is represented as $F = (Q, V, P, C)$, where Q denotes the set of quantifiers, used to express the existence of

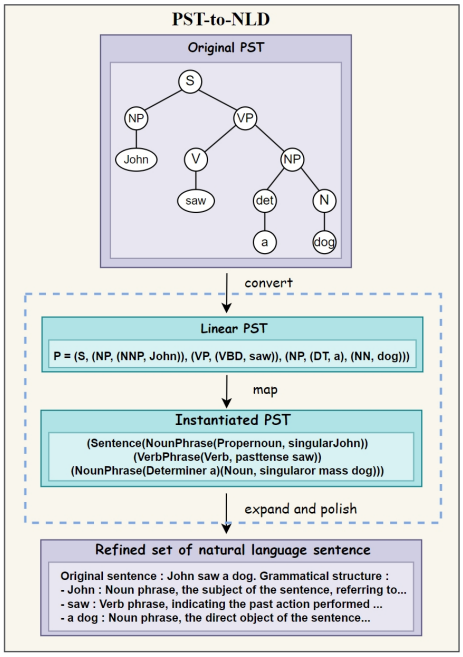


Figure 10: **The Whole process of Making PST-NLD.** The process of creating PST-NLD involves first converting the PST tree structure into a linear sequence of symbols using depth-first search (DFS). Then, a mapping function is applied to translate each node and its children into natural language descriptions. Finally, a language model is used to refine the generated descriptions, making them more natural and coherent.

variables, such as ∃ (exists) and ∀ (for all). V represents the set of variables, representing objects in FOL, typically denoted as x, y, z. P represents the set of predicates, used to express properties of objects or relationships between multiple objects. C represents the set of logical connectives, used to connect multiple propositions, including conjunction ($\wedge$), disjunction ($\vee$), and negation ($\neg$). An example of the original FOL structure is shown in the Figure 12.

Mapping FOL to Natural Language Descriptions. We define a mapping function $M = (D, L)$, where D is a set of symbol mappings that translates variables, predicates, and logical operators in FOL into natural language descriptions. L is a set of logical mapping rules that transforms the logical structure of FOL into natural language syntax. By applying these mapping rules to the initial FOL expressions, we can convert logical symbols into natural language descriptions.

Make_natural_pst_information

[system_prompt]
Given the provided Phrase Structure Tree (PST) and the interpretation for each tag, extract the original sentence and provide a simple explanation of its grammatical structure, including the role of each syntactic component.
###
Example 1:
Input:
PST: (TOP (S (ADVP (_ Indeed)) (_ ,) (NP (_ Republican) (_ lawyers))...)
Interpretation: (Top-level sentence structure (Sentence (Adverb Phrase (_ Indeed)) (_ ,) (Noun Phrase (_ Republican) (_ lawyers))...)
output:
original sentence:
Indeed, Republican lawyers identified only 300 cases of electoral fraud in the United States in a decade.
grammatical structure:
Indeed: Adverbial phrase, emphasizing the entire sentence.
Republican lawyers: Noun phrase, the subject of the sentence, referring to "Republican lawyers."
identified: Verb phrase, the action taken by the subject (to "identify").
only 300 cases: Direct object, specifying what was identified ("only 300 cases").
of electoral fraud: Prepositional phrase, describing the type of cases ("of electoral fraud").
in the United States: Prepositional phrase, indicating the location ("in the United States").
in a decade: Prepos
###
Example 2:
...
###
Example 3:
...

[user_prompt]
Input:
PST: `{pst_string}`
Interpretation: `{nl_interpretation}`
output:

Figure 11: The prompt of polishing sentence for making PST-NLD

Refinement of Natural Language Descriptions Using a Language Model. To ensure that the descriptions are coherent and fluent, we refine the generated descriptions S using the language model $F_{\text{LM}} : S \rightarrow S_{\text{refined}}$. The specific prompt is shown in the prompt (c) of Figure 13.

B Data Collection

B.1 The Process of Constructing Datasets for All Tasks of SR-LLM (training-free)

In this section, I will outline the process of collecting test data for the 10 tasks used in SR-LLM (training-free), including both the original text and three types of structured representations. The data statistics are summarized in the Table 1.

SNLI SNLI is a large and comprehensive dataset, with a test set containing 10,000 examples. Therefore, we directly used the test set for our experiments. The AMR, FOL, and PST

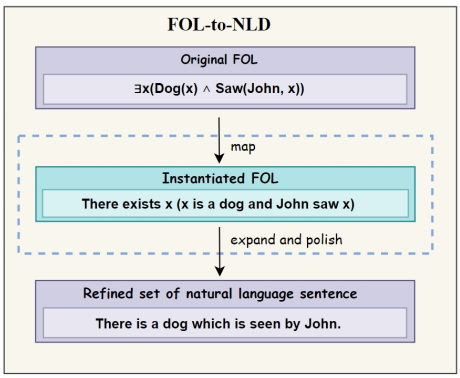


Figure 12: **The Whole process of Making FOL-NLD.** The process of converting FOL to NLD involves first mapping FOL symbols, such as variables, predicates, and logical operators, into natural language using predefined symbol mappings and logical rules. Then, the generated descriptions are refined using a language model to ensure they are coherent and fluent.

Table 1: Tasks and datasets used in SR-LLM (training-free)

Dataset	Task	Test Size
PAWS	Paraphrase Detection	8000
SNLI	Recognizing Textual Entailment	10000
WMT16	Translation	5999
CoNLL2003	Named Entity Recognition	3453
LOGIC	Logical Fallacy Detection	2449
SST-2	Sentiment Analysis	872
Pubmed45	Event Extraction	5000
WiC	Lexical Disambiguation	2038
SPIDER	Text2SQL Code Generation	8034
AGNEWS	Text Classification	7600

data were generated using GPT-4o-turbo in a few-shots setting, with the prompt provided in the Figure 14, Figure 15 and Figure 16.

CoNLL2003 CoNLL2003 is also a rich and complete dataset, with a test set of 3,453 examples, which we used directly. Structured representations were generated using the same method as described above.

SST-2 Since the official SST-2 test set does not contain labels, we used the full validation set of 872 examples as the test set for this experiment. Structured representations were generated using the same method as described above.

WiC The WiC test set consists of 1,400 examples, which is relatively small. Therefore, we combined the 648 examples from the validation set to create a larger test set. Structured representations were generated using the same method as described above.

Make_natural_fol_information

[system_prompt]
Given the First-Order Logic (FOL) formula and its symbolic interpretation, please generate a clear and human-understandable natural language sentence that accurately describes the meaning of this formula.
###
Example 1:
Input:
FOL: ∃x (Person(x) ∧ Doctor(x))
Interpretation: there exists x (Person(x) and Doctor(x))
output: There exists a person who is a doctor.
###
Example 2:
###
Example 3:
...

[user_prompt]
Input:
FOL: `{fol_string}`
Interpretation: `{nl_interpretation}`
output:

Figure 13: The prompt of polishing sentence for making FOL-NLD

Make_AMR

[system_prompt]
You are an NLP assistant whose task is to generate an Abstract Meaning Representation (AMR) for a given sentence.
The AMR should capture the essential meaning of the sentence in a structured format, representing the relationships between concepts in the sentence.
You must accurately represent the roles of entities, actions, and other relationships in the sentence.
Ensure that the representation is semantically complete and concise.
The AMR should not contain any extra explanations or interpretations, only the AMR structure.
###
Example 1:
Input Sentence: "The boy wants to eat an apple."
Output AMR:
(w / want-01
  :arg0 (b / boy)
  :arg1 (e / eat-01
    :arg0 b
    :arg1 (a / apple)))
###
Example 2:
...
###
Example 3:
...

[user_prompt]
Input Sentence: {sentence}
Output AMR:

Figure 14: The prompt of making AMR

AGNEWS AGNEWS is another large and comprehensive dataset, with a test set of 7,600 examples, which we used directly. Structured representations were generated using the same method as described above.

PAWS To ensure sufficient comparability in the experiments, the original text data and AMR representations for PAWS were sourced from Jin et al. (2024). And the FOL and PST representations were generated using the same method as described above.

Make_PST

[system_prompt]
You are an NLP assistant tasked with generating a Phrase Structure Tree (PST) for a given sentence. Your PST should illustrate the syntactic structure of the sentence, showing how words and phrases are hierarchically organized. Follow these guidelines:
Structure:Represent the sentence' s hierarchical structure with nodes for major constituents, such as noun phrases (NP), verb phrases (VP), adjective phrases (AdjP), adverbial phrases (AdvP), and prepositional phrases (PP).The tree should have a root node representing the entire sentence, typically labeled as S (Sentence).
Labeling:Label each node according to its syntactic function, such as NP for noun phrases, VP for verb phrases, and PP for prepositional phrases.Ensure that terminals (leaf nodes) represent individual words or word forms.
Hierarchy:Show the hierarchical relationship between constituents. For example, a verb phrase (VP) might be composed of a verb and a noun phrase (NP), where the VP node branches into its subcomponents. Include intermediate nodes where applicable, such as Determiners (Det) within noun phrases.
Ensure that the PST is detailed enough to accurately reflect the syntactic structure of the sentence.Avoid including extra explanations or interpretations; focus solely on the syntactic structure.
The pst should not contain any extra explanations or interpretations, only the pst structure.
###
Example 1:
Input Sentence: "The cat slept."
Output pst: '(S (NP (Det The) (N cat)) (VP (V slept)))'
###
Example 2:
...
###
Example 3:
...

[user_prompt]
Input:
FOL: {fol_string}
Interpretation: {nl_interpretation}
output:

Figure 15: The prompt of making PST

WMT16, LOGIC, Pubmed45, SPIDER The data collection for these tasks followed the same procedure as PAWS.

B.2 The Process of Constructing Datasets for All Tasks of SR-LLM (training-dependent)

In this section, I will explain the process of collecting both training and test data for the 10 tasks used in SR-LLM (training-dependent), including the original text and three types of structured representations. Data statistics are summarized in the Table 2.

Table 2: Tasks and datasets used in SR-LLM (training-dependent)

Dataset	Task	Train Size	Test Size
PAWS	Paraphrase Detection	10000	8000
SNLI	Recognizing Textual Entailment	10000	10000
WMT16	Translation	10000	5999
CoNLL2003	Named Entity Recognition	10000	3453
LOGIC	Logical Fallacy Detection	10000	2449
SST-2	Sentiment Analysis	10000	872
Pubmed45	Event Extraction	10000	5000
WiC	Lexical Disambiguation	5066	1048
SPIDER	Text2SQL Code Generation	7000	1034
AGNEWS	Text Classification	10000	7600

Make_FOL

[system_prompt]
You are an NLP assistant tasked with generating a First Order Logic (FOL) representation for a given natural language sentence. Please follow these steps:
Decompose the natural language sentence into statements and inferences.
Extract the entities, attributes, and relationships within the sentence.
Identify the relationships between entities and attributes using a reasoning model, incorporating real-world context.
Generate a First Order Logic formula that represents the logical structure of the sentence.
Ensure that the generated FOL expression accurately captures the logical structure of the sentence, reflects the relationships between entities, and contains no additional explanations or interpretative content.
The fol should not contain any extra explanations or interpretations, only the fol structure.
###
Example 1:
Input Sentence: "Alice loves Bob."
Output fol: 'Loves(Alice, Bob)'
###
Example 2:
...
###
Example 3:
...

[user_prompt]
Input Sentence: {sentence}
Output AMR:

Figure 16: The prompt of making FOL

PAWS, WMT16, Pubmed45, SNLI, CoNLL2003, SST-2, AGNEWS These datasets contain relatively large training sets. Therefore, we randomly selected 10,000 examples from each as the training set. The structured representations were generated using GPT-4o-turbo in a few-shot setting, with sample prompts provided in the figure. The test sets are the same as those used in the SR-LLM (training-free) experiments.

LOGIC Since the LOGIC dataset is relatively small, the training-free setup used all the available samples from the test, validation, and training sets combined, yielding a total of 2,449 samples as the test set. We retained these 2,449 samples for the test set in the training-dependent setting as well. To create the training set, we synthetically generated 10,000 logic examples using GPT-4o-turbo. The generation process is illustrated in the Figure 17, where a few-shot strategy was employed to guide the model to generate sentences containing different logical fallacies. The generated prompt is shown in Figures 18 and Figures 19. The type of logical error serves as the label, producing complete data points. Structured representations were generated in the same manner as described above.

SPIDER Since the official SPIDER test set is not publicly available, the training-free setup

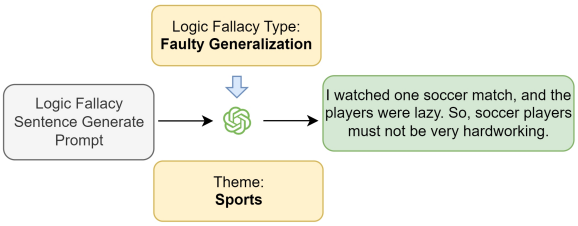


Figure 17: **The synthetic process for LOGIC data.** Taking the "Faulty Generalization" type as an example, we employed a few-shot strategy to guide the model in generating sentences containing the logical fallacy of "Faulty Generalization" To ensure greater sentence diversity, we incorporated a thematic element during generation, such as "Sports" as shown in the figure. This thematic addition helps produce a broader variety of sentence while maintaining the specific logical error, leading to a richer and more varied dataset.

used a combination of training and validation sets as the test set. However, due to the complexity of generating SPIDER-like data, we used the original 7,000 training examples for the training set in the training-dependent setting and the 1,034 validation examples as the test set. Structured representations were generated as described above.

WiC As the WiC training set is relatively small, we combined the 648 validation examples with the original training set to create a total of 5,066 training samples. Structured representations were generated using the same method as described above.

Logic Fallacy Sentence Generate Prompt (a)

[system_prompt]

Faulty_generalization_prompt
You are an NLP assistant whose task is to generate a sentence that demonstrates the "Faulty Generalization" fallacy.
This fallacy occurs when a conclusion is drawn about a whole group based on a sample that is too small or unrepresentative.
###
Example:
input theme: "travel"
output sentence: I visited two countries, and both had rude people. Therefore, all countries must have rude people.

Faulty_causality_prompt
You are an NLP assistant...that demonstrates the "False Causality" fallacy.
This fallacy occurs when someone assumes that because two events occur together or in sequence, one must cause the other.
###
Example:
input theme: "exercise"
output sentence: Ever since I started running in the mornings, it hasn't rained. Running must stop the rain.

Circular_claim_prompt
You are ... "Circular Claim" fallacy.
This fallacy occurs when the conclusion of an argument is used as a premise to support itself, creating a loop without proving the claim.
###
Example:
...

ad_populum_prompt
You are ... "Ad Populum" fallacy.
This fallacy occurs when an argument is made based on the belief that if many people believe something, it must be true.
###
Example:
...

ad_hominem_prompt
You are ... "Ad Hominem" fallacy.
This fallacy occurs when someone attacks the person making the argument rather than addressing the argument itself.
###
Example:
...

deductive_fallacy_prompt
You are ... "Deductive Fallacy" fallacy.
This fallacy occurs when the premises of an argument do not logically lead to the conclusion, even if the premises are true.
###
Example:
...

[user_prompt]
input theme: {theme}
output sentence

Figure 18: Logic Fallacy Generate Prompt (a)

C Additional Experiments

C.1 Comparative Analysis of Different SR Combinations and Their Impact on LLM Reasoning

We conducted an in-depth comparison of the performance of different structured representations (SR) and explored their combinations to assess whether joint usage could further enhance LLM reasoning capabilities. Figure 20 summarizes the average performance improvements across all tasks. The results indicate that the use of individual SRs such as AMR, PST, and FOL did not lead to significant performance enhancements, which is consistent with the findings of (Jin et al., 2024). Moreover, when multiple SRs were introduced simultaneously, their combined complexity posed additional challenges for the LLMs, further dispersing the model's attention and resulting in poorer performance compared to using a single SR. In contrast, when relatively weaker LLMs were provided with more comprehensible semantic features (AMR) and logical features (FOL), their average performance improved. The integration of these two types of features complemented each other, leading to better overall results. However, the contribution of syntactic features (PST) was relatively less effective and, in some cases, even negated the positive effects of semantic and logical features.

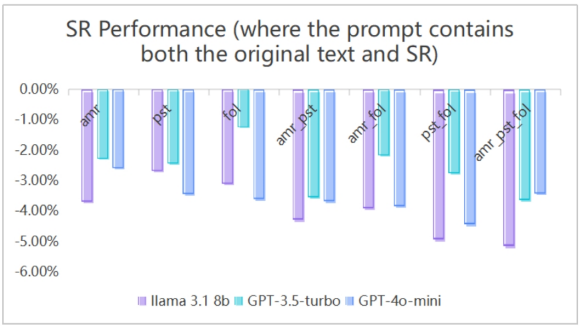
C.2 Optimal Text-to-SR Ratio Analysis

To further investigate the most optimal ratio of between G(text) and G(SR), I selected five tasks, which includes PAWS, LOGIC, Pubmed45, SPIDER, WMT16 for additional

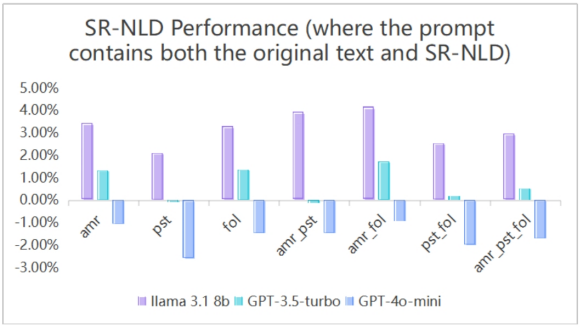
Logic Fallacy Sentence Generate Prompt (b)

```
[system_prompt]

# appeal_to_emotion_prompt
You are ... "Appeal to Emotion" fallacy.
This fallacy occurs when an argument is made by manipulating
emotions rather than using logical reasoning.
###
Example:
...

# false_dilemma_prompt
You are ... "False Dilemma" fallacy.
This fallacy occurs when only two options are presented as if they
are the only possible choices, when in fact others exist.
###
Example:
...

# equivocation_prompt
You are ... "Equivocation" fallacy.
This fallacy occurs when a key term in an argument is used in an
ambiguous way, with one meaning in one part of the argument and
another meaning in another part.
###
Example:
...
# fallacy_of_extension_prompt
You are ... "Fallacy of Extension" fallacy.
This fallacy occurs when an argument is extended beyond its
original scope in an irrelevant way, or when the argument
misrepresents the opposing viewpoint to weaken it.
###
Example:
...

# fallacy_of_relevance_prompt
You are ... "Fallacy of Relevance" fallacy.
This fallacy occurs when an argument's premises are irrelevant to
the conclusion.
###
Example:
...

# fallacy_of_credibility_prompt
You are ... "Fallacy of Credibility" fallacy.
This fallacy occurs when an argument is based on the credibility of
the source rather than on logical reasoning or evidence.
###
Example:
...

# intentional_fallacy_prompt
You are ... "Intentional Fallacy" fallacy.
This fallacy occurs when someone interprets a text or statement
based on the author's intentions rather than on the content of the
text or statement itself.
###
Example:
...

[user_prompt]
input theme: {theme}
output sentence
```

Figure 19: Logic Fallacy Generate Prompt (b)



(a) Average performance of SR Performance in different Tasks



(b) Average performance of SR-NLD Performance in different Tasks

Figure 20: **Performance comparison of different SR combinations.** (a) The average performance enhancement (Δ), for various SR combinations across different tasks. (b) The average performance enhancement (Δ), for different SR-NLD combinations across various tasks.

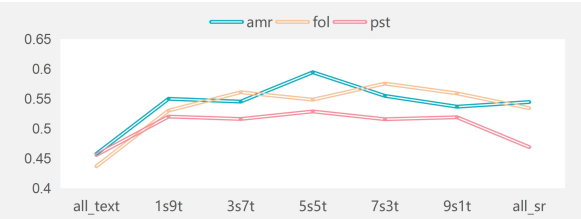


Figure 21: Comparison of average performance of models at different scales in all tasks.

experiments, adjusting the ratio of text to structured representations in the Gen-SR dataset to identify the optimal balance. The experimental results are shown in the Figure 21. As can be observed, the fluctuations in performance with different ratios are relatively small. For both AMR and PST, a 50-50 ratio between text and structured representations appears to be the most effective. However, for FOL, a 30-70 ratio (whether favoring structured representations) yields better results. This is a preliminary exploration, and I believe it represents a promising direction for further research.

C.3 Enhancing LLM's Understanding of SR during Pretraining.

We further conducted experiments during the pretraining phase with the goal of enhancing LLM's ability to comprehend structured representations, aiming for performance improvements in downstream tasks. Specifically, we collected 1GB of task-agnostic SR data, including AMR, PST, and FOL, following a similar procedure as in previous data collection efforts, and applied this data to the pre-training of the Llama3.1-8B-Instruct model. Building on this, we further conducted SFT, the same as SR-LLM (training-dependent), on five datasets. The final average performance results are shown

in the Table 3.

The experimental results show that, compared to the vanilla model without pre-training, the pre-trained model indeed exhibited performance improvements in downstream tasks, though the improvements were relatively modest, with an average increase of less than 1%. However, after applying SFT on the pre-trained model, its performance was actually inferior to that of the vanilla model trained directly with SFT. We hypothesize that this phenomenon may be due to the model forming certain inherent understandings of structured representations during the pre-training phase, which hindered its ability to establish effective connections between structure and tasks during SFT, leading to worse performance compared to the vanilla model. This phenomenon highlights a potential conflict in how the model processes structured information during the pre-training and fine-tuning phases, which warrants further exploration and resolution in future research.

Table 3: **The SR enhancement of models with different training strategies.** These are the average SR Enhancement results across all tasks under different training strategies. Green indicates the best performance within the same SR, while red represents the worst performance.

AMR	FOL	PST	Pretrain	SFT	Δ
✓					**-3.51%**
✓			✓		0.56%
✓			✓	✓	-1.16%
✓				✓	**11.59%**
	✓				**-2.83%**
	✓		✓		1.30%
	✓		✓	✓	3.10%
	✓			✓	**6.45%**
		✓			**-3.61%**
		✓	✓		-0.18%
		✓	✓	✓	1.58%
		✓		✓	**2.91%**

D Examples of Gen-SR

We present specific examples of Gen-SR in this section. Figure 22 shows an example of G(text), Figure 23 shows an example of G(AMR), Figure 24 shows an example of G(PST), and Figure 25 shows an example of G(FOL).

G(text)

[Instruction]

You are an advanced natural language understanding model designed for the task of Sentiment Analysis.

Your goal is to classify the sentiment of a given sentence.

You will classify each sentence into one of the following two categories:

1) Negative: The sentiment of the sentence is negative.

2) Positive: The sentiment of the sentence is positive.

You will be provided with a sentence. Based on the content of the sentence, classify the sentiment as either positive or negative.

Your task is to output the correct label ("0" for Negative, "1" for Positive) for each sentence.

Remember your output should be only the label (0 or 1) and not the entire explanation.

[Input]

Input Sentence: `'It's nothing interesting in unfaithful whatsoever'`

[Output]

Output Responsece: `'0'`

Figure 22: The Example of G(text)

G(AMR)

[Instruction]

You are an advanced natural language understanding model designed for the task of Sentiment Analysis.

Your goal is to classify the sentiment of a given sentence.

You will classify each sentence into one of the following two categories:

1) Negative: The sentiment of the sentence is negative.

2) Positive: The sentiment of the sentence is positive.

You will be provided with a sentence and its Abstract Meaning Representation (AMR).

The AMR will help you understand the sentence's semantic structure, including the key participants and actions.

Based on the sentence and its AMR, classify the sentiment as either positive or negative.

Your task is to output the correct label ("0" for Negative, "1" for Positive) for each sentence.

Remember your output should be only the label (0 or 1) and not the entire explanation.

[Input]

Input Sentence: `'It's nothing interesting in unfaithful whatsoever'`

Input AMR: `'(neg / nothing\n  :mod (i / interesting)\n  :arg1 (u / unfaithful))'`

[Output]

Output Responsece: `'0'`

Figure 23: The Example of G(AMR)

E Prompt of Testing the SR-LLM

We present the complete prompts for our experiments, including both CoT and One-shot examples, using the SNLI dataset as an illustration in Figures 26, Figures 27 and Figures 28.

G(PST)

[Instruction]

You are an advanced natural language understanding model designed for the task of Sentiment Analysis.

Your goal is to classify the sentiment of a given sentence.

You will classify each sentence into one of the following two categories:

1) Negative: The sentiment of the sentence is negative.

2) Positive: The sentiment of the sentence is positive.

You will be provided with a sentence and its Phase Structure Tree (PST).

The PST will help you analyze the grammatical structure of the sentence, identifying modifiers, adjectives, and sentiment-bearing phrases.

Based on the sentence and its PST, classify the sentiment as either positive or negative.

Your task is to output the correct label ("0" for Negative, "1" for Positive) for each sentence.

Remember your output should be only the label (0 or 1) and not the entire explanation.

[Input]

Input Sentence: **'It's nothing interesting in unfaithful whatsoever'**

Input AMR: **'(NP (AdjP (Adj warm) (Conj and) (Adj winning)) (N central) (N performance))'**

[Output]

Output Responsece: **'0'**

Figure 24: The Example of G(PST)

G(FOL)

[Instruction]

You are an advanced natural language understanding model designed for the task of Sentiment Analysis.

Your goal is to classify the sentiment of a given sentence.

You will classify each sentence into one of the following two categories:

1) Negative: The sentiment of the sentence is negative.

2) Positive: The sentiment of the sentence is positive.

You will be provided with a sentence and its First-Order Logic (FOL) representation.

The FOL will help you understand the logical structure and any relationships that contribute to the overall sentiment.

Based on the sentence and its FOL, classify the sentiment as either positive or negative.

Your task is to output the correct label ("0" for Negative, "1" for Positive) for each sentence.

Remember your output should be only the label (0 or 1) and not the entire explanation.

[Input]

Input Sentence: **'It's nothing interesting in unfaithful whatsoever'**

Input FOL: **'¬∃x (Interesting(x) ∧ Unfaithful(x))'**

[Output]

Output Responsece: **'0'**

Figure 25: The Example of G(FOL)

Prompt COT:
snli

[system_prompt]
You are an advanced natural language understanding model designed for the task of Recognizing Textual Entailment (RTE).
Your goal is to determine the relationship between two sentences: a premise and a hypothesis.
You will classify each pair of sentences into one of the following three categories:

1) Entailment: The hypothesis can logically be inferred from the premise.
2) Contradiction: The hypothesis directly contradicts the premise.
3) Neutral: The hypothesis is neither entailed nor contradicted by the premise.

You will be provided with the {AMR(nld)/ PST(nld)/ FOL(nld)} for both the premise and the hypothesis.
Your task is to analyze the semantic structure using the {AMR(nld)/ PST(nld)/ FOL(nld)} and output the correct label.

Remember, your output should be the label number only, without any additional explanation.

[user_prompt]
Premise: {premise}
Hypothesis: {hypothesis}
optional
Input AMR Premise: {amr_p}
Input AMR Hypothesis: {amr_h}

Input PST Premise: {pst_p}
Input PST Hypothesis: {pst_h}

Input FOL Premise: {fol_p}
Input FOL Hypothesis: {fol_h}

Input AMR Premise description: {amr_p_nld}
Input AMR Hypothesis description: {amr_h_nld}

Input PST Premise description: {pst_p_nld}
Input PST Hypothesis description: {pst_h_nld}

Input FOL Premise description: {fol_p_nld}
Input FOL Hypothesis description: {fol_h_nld}
Output:

Figure 26: The COT prompt of SNLI

Prompt One-Shot:
snli

[system_prompt]
You are an advanced natural language understanding model designed for the task of Recognizing Textual Entailment (RTE). Your goal is to determine the relationship between two sentences: a premise and a hypothesis.
You will classify each pair of sentences into one of the following three categories:

1) Entailment: The hypothesis can logically be inferred from the premise.
2) Contradiction: The hypothesis directly contradicts the premise.
3) Neutral: The hypothesis is neither entailed nor contradicted by the premise.

You will be provided with {AMR(nld)/ PST(nld)/ FOL(nld)} for both the premise and the hypothesis.
Your task is to analyze the {semantic structure/ grammatical structure/ logical relationship} using the {AMR(nld)/ PST(nld)/ FOL(nld)} and output the correct label ("0" for Entailment, "1" for Contradiction, and "2" for Neutral).

Remember, your output should be the label number only, without any additional explanation.

###
Example:
[snli_example]

[user_prompt]
Premise: {premise}
Hypothesis: {hypothesis}
optional
Input AMR Premise: {amr_p}
Input AMR Hypothesis: {amr_h}

Input PST Premise: {pst_p}
Input PST Hypothesis: {pst_h}

Input FOL Premise: {fol_p}
Input FOL Hypothesis: {fol_h}

Input AMR Premise description: {amr_p_nld}
Input AMR Hypothesis description: {amr_h_nld}

Input PST Premise description: {pst_p_nld}
Input PST Hypothesis description: {pst_h_nld}

Input FOL Premise description: {fol_p_nld}
Input FOL Hypothesis description: {fol_h_nld}
Output:

Figure 27: The One-Shot prompt of SNLI

Prompt One-Shot:
snli_example

[snli_example]
 # text
Premise: "Four adults eat while sitting on a tile floor."
Hypothesis: "A group of people eat food."
 # amr
Input AMR Premise: '(e / eat-01\n :arg0 (a / adult :quant 4)\n :arg1 (s / sit-01\n :arg0 a\n :arg1 (f / floor\n :mod (t / tile))))'
Input AMR Hypothesis: '(e / eat-01\n :arg0 (g / group\n :arg1 (p / person))\n :arg1 (f / food))'
 # pst
Input PST Premise: '(S (NP (Det Four) (N adults)) (VP (V eat) (SBAR (S (AdvP (Adv while)) (VP (V sitting) (PP (P on) (NP (Det a) (N tile) (N floor))))))))'
Input PST Hypothesis: '(S (NP (Det A) (N group) (PP (P of) (NP (N people)))) (VP (V eat) (NP (N food))))'
 # fol
Input FOL Premise: '∃ x ∃ y (Adult(x) ∧ Adult(y) ∧ Adult(z) ∧ Adult(w) ∧ Eating(x) ∧ Eating(y) ∧ Eating(z) ∧ Eating(w) ∧ Sitting(x) ∧ Sitting(y) ∧ Sitting(z) ∧ Sitting(w) ∧ TileFloor(Floor) ∧ On(x, Floor) ∧ On(y, Floor) ∧ On(z, Floor) ∧ On(w, Floor))'
Input FOL Hypothesis: '∃ x (Group(x) ∧ People(x) ∧ ∃ y (Food(y) ∧ Eat(x, y)))'
 # amr_nld
Input AMR Premise description: '0 Four adults are the ones eating. \n1 There are four adults participating in the activity. \n2 Eating involves them sitting in a position. \n3 Each adult is sitting while they eat. \n4 They are seated on a floor. \n5 The floor is described as being tiled. '
Input AMR Hypothesis description: '0 A group is responsible for the action of eating. \n1 This group consists of people. \n2 The food is what they are consuming.'
 # pst_nld
Input PST Premise description: 'Original sentence:\nFour adults eat sitting on a tile floor.\n\nGrammatical structure:\n- Four adults: Noun phrase, the subject of the sentence, indicating who is performing the action (the \"adults\").\n- eat: Verb, the main action of the sentence performed by the subject.\n- sitting on a tile floor: Subordinate clause acting as an adverbial phrase, providing additional information about how the action of eating is being carried out.\n - sitting: Gerund, indicating the manner of the action (how they are eating).\n - on a tile floor: Prepositional phrase, describing the location of the action.\n - on: Preposition, introducing the prepositional phrase.\n - a tile floor: Noun phrase, the object of the preposition, specifying where the action is taking place (the \"tile floor\").'
Input PST Hypothesis description: 'Original sentence:\nA group of people eat food.\n\nGrammatical structure:\n- A group of people: Noun phrase, the subject of the sentence, indicating the group performing the action (the \"people\").\n- eat: Verb, the main action performed by the subject.\n- food: Noun phrase, the object of the action, specifying what is being eaten.'
 # fol_nld
Input FOL Premise description: 'The child is on the library steps.'
Input FOL Hypothesis description: 'There exists a child who is on the steps inside the library.'
 # output
Output: 0

Figure 28: The One-Shot prompt of SNLI's example