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

Anonymous ACL submission

Abstract

Structured representations, exemplified by Abstract Meaning Representation (AMR), have long been pivotal in computational linguistics. However, their role remains ambiguous in the Large Language Models (LLMs) era. Initial attempts to integrate structured representation into LLMs via a 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. Consequently, we propose SR-LLM, an innovative framework with two settings to explore a superior way of integrating structured representation with LLMs from training-free and training-dependent perspectives. The former integrates structural information through natural language descriptions in LLM prompts, whereas its counterpart augments the model's inference capability through fine-tuning on linguistically described structured representations. Performance improvements were observed in widely downstream datasets, with particularly notable gains of 3.17% and 12.38% in PAWS. To the best of our knowledge, this work represents the pioneering demonstration that leveraging structural representations can substantially enhance LLMs' inference capability. We hope that our work sheds light and encourages future research to enhance the reasoning and interoperability of LLMs by structure data.

1 Introduction

Structured representations (SR), manifested in 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), have been fundamental to NLP (Manning, 1999; Collobert et al., 2011), serving as sophisticated frameworks for capturing semantic relationships and linguistic structures (Banarescu et al., 2013; Wang et al., 2015). An example of AMR, PST, and FOL is depicted in Figure 2.

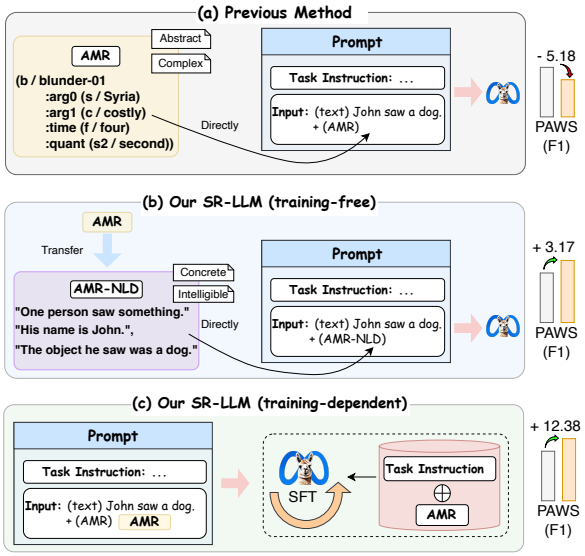


Figure 1: We propose two novel AMR integration approaches: a training-free method using natural language descriptions and a training-dependent fine-tuning paradigm. Evaluation on PAWS shows +3.17% and +12.38% improvements respectively, contrasting with the -5.18% decline in conventional code-format methods.

In the era of LLMs, the paradigm for optimal SR integration remains an open research challenge. Despite LLMs' capabilities, direct integration of SR into prompts, as illustrated in Figure 1, has proven counterproductive (Jin et al., 2024). We posit that this performance degradation stems from LLMs' inherent limitations in processing structured representations, where direct exposure to complex linguistic structures impedes rather than enhances their reasoning process.

To address the aforementioned challenges and effectively leverage SR in LLMs, we in-

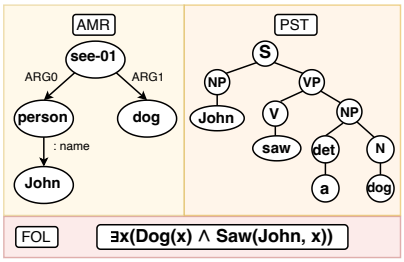


Figure 2: The AMR, PST, and FOL of the sentence "John saw a dog".

troduce SR-LLM, a comprehensive framework with dual configurations for structural knowledge integration. The training-free approach transforms SR into natural language descriptions (SR-NLD), enhancing prompt comprehension by reformulating structured information into semantically rich, accessible formats that facilitate nuanced reasoning and reduce ambiguity. Complementarity, the training-dependent paradigm employs supervised fine-tuning on task-specific SR datasets (termed Gen-SR) to establish robust SR-task connections through iterative exposure to structured data, enabling the model to develop sophisticated internal representations and leverage deep structural knowledge during inference across diverse NLP tasks.

Our empirical evaluation encompasses a comprehensive suite of NLP benchmarks, spanning diverse linguistic phenomena from paraphrase detection (Mihalcea et al., 2006; Dolan and Brockett, 2005) and textual entailment recognition (Dagan et al., 2005; Bowman et al., 2015) to machine translation (Bahdanau, 2014; Johnson et al., 2017). This diverse benchmark selection enables rigorous evaluation of our methods across the NLP spectrum. Experimental results demonstrate the superiority of our methods over existing approaches: on PAWS, while conventional method exhibits a 5.18% performance degradation, our training-free and training-dependent approaches achieve +3.17% and +12.38% improvements respectively, which validating the efficacy of our structured information integration paradigm.

Our contributions are as follows:

- We introduce SR-LLM, a novel framework that facilitates SR integration with LLMs through dual paradigms: training-free adaptation and supervised fine-tuning.
- We provide insights into how different types of SR (AMR, PST, FOL) impact LLMs performance across various tasks.
- To the best of our knowledge, we are the first to show that combining such SR does in fact improve LLM performance, which opens up new avenues for enhanced 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 $X = (x_1, x_2, \ldots, x_n)$, where $x_i \in V$ represents a token drawn from the vocabulary V, we also introduce the structured representation Z. Z serves as auxiliary information derived from X and can take various forms, such as AMR, PST, or FOL. These SRs capture semantic, syntactic, or logical information and provide complementary insights to natural language understanding.

The task involves generating an output sequence $Y = (y_1, y_2, \ldots, y_m)$, where each y_i belongs to either the target vocabulary or a structured semantic output space. This transformation is performed by a model f, defined as:

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

Here, f specifies how X and Z are utilized to complete a specific task by integrating natural language input with its structured representation.

The primary goal of this research is to optimize the definition of f to achieve the most effective use of X and Z, thereby maximizing task performance. Specifically, the objective is to identify the optimal model f^* that maximizes the evaluation metric $P(\cdot)$, such as accuracy or F1 score:

$$f^* = \arg\max_f P(f(X, Z)) \tag{2}$$

3 Method

This chapter introduces the SR-LLM framework, a novel paradigm designed to investigate the efficacious integration of SR into

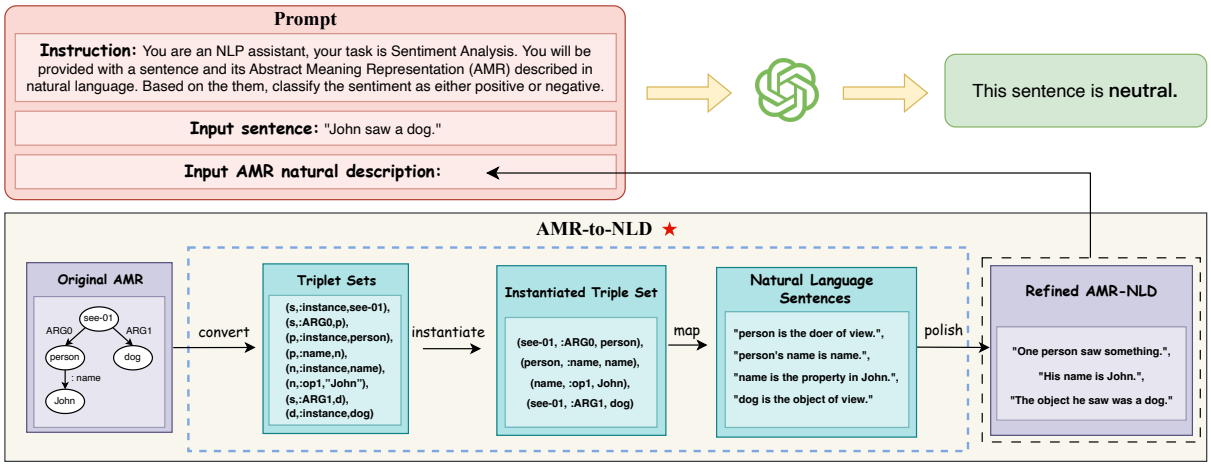


Figure 3: **The whole process of SR-LLM in training-free setting**. Initially, a task-specific prompt consists of an instruction, input sentence, and input SR structure (AMR is used here). Subsequently, the original AMR undergoes transformation via the **AMR-to-NLD** module, which employs predefined 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.

LLMs. The SR-LLM framework encompasses 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.

3.1 SR-LLM Training-Free

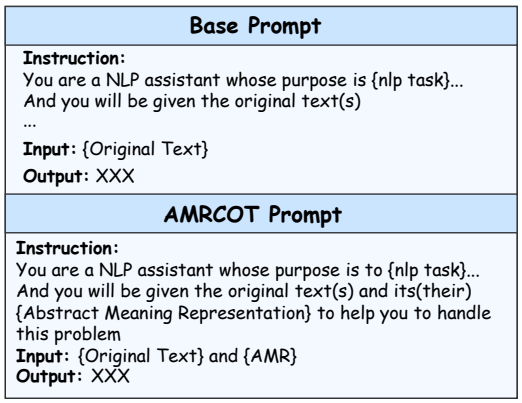


Figure 4: **Base prompt and AMRCOT prompt. (Top)** This is the original task prompt, with only the raw text as input, serving as the standards for performance. **(Bottom)** This is the AMRCOT prompt method proposed by Jin et al. (2024), serving as a baseline.

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 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. 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. Finally, these descriptions are incorporated into the prompt and input into the target LLM. 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.

Next, we shall elucidate the SR-to-NLD process, employing AMR-NLD as our quintessential exemplar, which shown in the Algorithm 1.

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}

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 GPT-4o Mini to produce coherent AMR-NLD. To mitigate the risk of hallucination, we implemented a voting mechanism based on multiple generations. 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.1 for completeness. Different from traditional SR-to-Text approaches, which generate a structurally coherent and fluent text based on the SR, such as the "input sentence" in Figure 4. SR-to-NLD aims to collaboratively describe the structured information through multiple sentences, as illustrated by the Refined AMR-NLD in Figure 4.

3.2 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 parts: one consists of task-specific instruction pairs based on original text, while the other adds SRs 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 of these two are shown in the Appendix D. This mixed approach allows LLM to not only learn instruction-following for downstream tasks from G(text), but also to establish more robust connections between tasks and structures from G(SR), making the model achieve more effective improvements compared to learning solely from text.

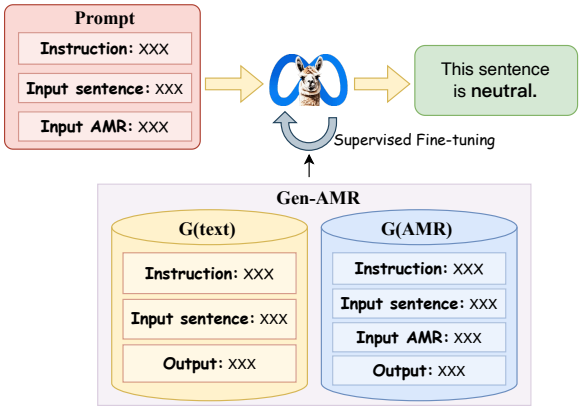


Figure 5: **The whole process of SR-LLM in training-dependent setting**. Taking AMR as an example, a dataset called **Gen-AMR**, created by combining inputs consisting of sentences and their corresponding AMR structures, is utilized for the SFT of LLM to enhance the reasoning capability.

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),

Table 1: **Performance of SR-LLM(training-free).** In the table, a checkmark under "SR" indicates that the original SR was added to the prompt, while a checkmark under "SR-NLD" (highlighted with a gray background) represents the inclusion of SR-NLD in the prompt, which corresponds to the results of SR-LLM (training-free). No checkmarks indicate the use of the original prompt, serving as the control group for comparison. Our focus is on the performance differences between adding SR and SR-NLD, as well as their respective differences compared to the control group.

SR	SR-NLD (Ours)	PAWS (F1)	Logic (F1)	Pubmed45 (F1)	AGNEWS (F1)	WiC (F1)	SNLI (F1)	CoNLL2003 (F1)	SST-2 (F1)	WMT16 (BLEU)	SPIDER (F1)
					(a) Llama3.1- 8b-Instruct						
		41.59	15.48	24.35	53.88	43.99	25.81	46.28	68.72	13.16	24.80
✓		36.41	14.20	20.69	48.17	42.05	23.17	41.75	65.66	12.34	21.53
	✓	**44.77**	**18.27**	**26.10**	**56.67**	**48.17**	**28.87**	**48.73**	**71.77**	**14.10**	**29.60**
					(b) GPT 3.5-turbo						
		56.94	38.63	27.14	**85.12**	50.61	38.93	**56.52**	90.46	26.13	39.63
✓		56.10	36.27	25.63	81.33	51.60	32.00	54.67	86.90	25.77	39.07
	✓	**57.97**	**39.40**	**28.17**	84.07	**55.27**	**41.47**	55.17	**92.60**	**27.07**	**42.27**
					(c) GPT 4o-mini						
		75.80	**48.10**	**38.65**	**85.26**	**58.47**	40.59	**65.27**	91.39	**26.80**	41.55
✓		73.50	47.32	33.11	81.62	46.65	41.30	59.21	91.01	26.21	39.33
	✓	**76.48**	47.95	36.66	83.45	56.63	**42.00**	64.12	**92.83**	26.76	**43.57**

and AGNEWS for text classification (Zhang et al., 2015).

Regarding the source of SR datasets, we used a dual-source strategy: one part includes high-quality AMR datasets from Jin (Jin et al., 2024), covering five core tasks; the other is automatically constructed using GPT-4o, comprising supplementary AMR, PST, and FOL data. The detailed collection processes and results provided in the Appendix B.1.

4.2 Training-Free Results

Experimental Details. We conducted experiments on the Llama3.1-8b-Instruct (Dubey et al., 2024), GPT-3.5-turbo, and GPT-4o-mini (Achiam et al., 2023) models, arranged from weak to strong according to 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. All experiments were conducted independently on three types of SRs: AMR, FOL, and PST. Both PST and FOL were incorporated into the prompts using the same approach as AMR-COT (Jin et al., 2024). For brevity, the results obtained from these experiments were averaged and presented. For detailed prompts, refer to the Appendix E.

Result Analysis. First, as shown in Table 1, incorporating SR-NLD into the prompt consistently outperforms incorporating the original SR. This indicates that for LLMs, transforming abstract SRs into natural language formats more familiar to the models is an effective strategy for enhancing their ability to interpret and apply structured information. Meanwhile, the comparision of the three models also reveals that the gradual decrease in the benefit of structured information as model performance increases. Specifically, for the Llama3.1-8b-Instruct model, results with SR-NLD significantly and consistently surpass those of the original prompt (i.e., without SR or SR-NLD). For GPT-3.5-turbo, most results show improvement, whereas for GPT-4o-mini, approximately half of the results demonstrate improvement, albeit with a smaller margin. 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.

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, more detailed experimental parameters can be found in the Appendix A.2. The whole process of fine-tuning is a joint training across data from 10 tasks, rather than task-specific fine-tuning for any

Table 2: **Performance of SR-LLM(training-dependent).** G(text) and G(SR) represent the types of training data, with 50% and 10% indicating their respective proportions in the total training dataset. Our focus is on the best performance of the model across various tasks under different fine-tuning strategies, as well as the performance differences between adding SR and the control group.

FT Strategy	SR	PAWS (F1)	Logic (F1)	Pubmed45 (F1)	AGNEWS (F1)	WiC (F1)	SNLI (F1)	CoNLL2003 (F1)	SST-2 (F1)	WMT16 (BLEU)	SPIDER (EM)
-		41.59	15.48	24.35	53.88	43.99	25.81	46.28	68.72	13.16	24.80
	✓	36.41	14.20	20.69	48.17	42.05	23.17	41.75	65.66	12.34	21.53
100% G(text)		68.94	26.21	78.91	76.52	66.97	35.53	75.79	75.59	29.07	41.20
	✓	64.07	16.84	77.33	67.14	67.05	35.36	71.73	74.65	28.41	38.47
100% G(SR)		65.34	25.23	81.13	75.10	66.44	36.68	75.40	77.49	26.93	37.07
	✓	75.39	29.89	**82.02**	81.99	70.82	**56.62**	76.27	81.62	**30.80**	40.60
50% G(SR)		68.66	26.77	79.78	75.77	69.48	36.49	75.42	77.13	26.14	42.40
+ 50% G(text)	✓	**81.04**	**36.52**	81.85	**82.63**	**74.68**	54.92	**76.67**	**83.72**	30.33	**48.93**

single dataset. Detailed data collection procedures and specific training data configurations are provided in the Appendix B.2. To provide a comparative analysis, we conducted three sets of experiments using the following datasets: 100%G (text), 100%G (SR), and a 50%G (text) mixed with 50% G (SR). The 50%-50% ratio was chosen because we considered it to be the most balanced approach. Further experiments, elaborated in Appendix C.2, also confirmed that this is the optimal mixing ratio. And we employed a random sampling approach. All experiments were conducted independently on three types of SRs and for brevity, the results obtained from these experiments were averaged and presented.

Result Analysis. As shown in the Table 2, when the fine-tuning dataset includes a certain proportion of SRs and incorporates SRs in the prompt, the model achieves superior performance in downstream tasks, consistently surpassing the case where the training data consists solely of text. Additionally, we observe that models fine-tuned with SRs data perform significantly better with prompts that include SRs, compared to the original prompts without SR. Conversely, when the training data consists entirely of text, the opposite trend is observed. These findings suggest that when a model establishes a strong association between tasks and structured representations during training, it can leverage this information more effectively during inference. Furthermore, when the training data is entirely composed of structured representations, the performance is inferior to that achieved with a balanced mix of text and structured data. This highlights the critical importance of a balanced integration of raw text and structured representations in maximizing the model's reasoning capabilities.

Table 3: **Performance between different AMR Source.** Each data represents the performance difference of the model when using AMRs generated by GPT-4o versus AMRBART, calculated as the performance of AMRBART minus that of GPT-4o. As shown, the differences are almost all below 1%.

AMR	AMR (NLD)	PAWS (F1)	Logic (F1)	Pubmed45 (F1)	WMT16 (BLEU)	SPIDER (EM)
(a) Llama3.1-8b-Instruct						
✓		0.40	-0.07	0.01	0.13	0.28
	✓	0.77	-0.13	0.50	-0.02	-0.01
(b) GPT 3.5-turbo						
✓		0.45	0.57	-0.15	0.08	0.12
	✓	0.02	-2.40	0.52	0.23	0.21
(c) GPT 4o-mini						
✓		0.08	0.07	0.53	0.49	0.02
	✓	-0.11	0.61	0.61	-0.13	-0.13

4.4 Auxiliary Validation Experiments

SR from High-Quality SR-Parsing Model. To validate the reliability of the generated SRs, we choose AMRBART (Bai et al., 2022) to generate the required AMRs, and experiments were conducted to compare the results with those generated by GPT-4o. It is a model that demonstrates exceptional performance in the AMR parsing domain with a Smatch score of 85.4 on the AMR Parsing Leaderboard, ranking among the top-performing models. As shown in the Table 3, the performance differences between AMRs and AMR-NLDs derived from these two sources were minimal, almost always within 0.5%. This indicates that the quality of the AMRs produced by AMRBART is comparable to those generated by our method.

Table 4: **Performance between different AMR Quality**. The numbers in parentheses represent the performance differences between adding AMR or AMR-NLD and the control group. 'Flawed' means the AMR is ambiguous or structurally flawed. 'Gold' means the AMR is double checked by human and LLM.

AMR Quality	AMR	AMR-NLD	PAWS (F1)	Logic (F1)	Pubmed45 (F1)	WMT16 (BLEU)	SPIDER (EM)
			(a) Llama3.1-8b-Instruct				
-	-	-	42.19	14.32	23.67	13.66	22.58
Flawed	✓		34.5 (-7.69)	11.52 (-2.8)	19.41 (-4.26)	11.07 (-2.59)	18.26 (-4.32)
Gold	✓		42.48 (+0.29)	14.7 (+0.38)	23.43 (-0.24)	14.65 (+0.99)	22.93 (+0.35)
Flawed		✓	32.91 (-9.29)	11.56 (-2.76)	18.39 (-5.28)	11.06 (-2.6)	18.49 (-4.09)
Gold		✓	**46.96 (+4.76)**	**18.98 (+4.66)**	**28.62 (+4.95)**	**19.13 (+5.47)**	**28.02 (+5.44)**
			(b) GPT 3.5-turbo				
-	-	-	56.04	43.79	28.29	26.01	40.28
Flawed	✓		51.57 (-4.47)	41.58 (-2.21)	25.71 (-2.58)	23.79 (-2.22)	36.66 (-3.62)
Gold	✓		54.53 (-1.51)	44.7 (+0.91)	29.47 (+1.19)	26.17 (+0.15)	39.77 (-0.51)
Flawed		✓	51.33 (-4.71)	39.79 (-4.01)	26.9 (-1.38)	24.37 (-1.64)	36.74 (-3.54)
Gold		✓	**56.78 (+0.74)**	**46.49 (+2.7)**	**32.0 (+3.71)**	**28.72 (+2.71)**	**44.81 (+4.53)**
			(c) GPT 4o-mini				
-	-	-	68.71	44.95	37.07	29.02	40.05
Flawed	✓		65.63 (-3.08)	42.74 (-2.2)	35.42 (-1.66)	27.31 (-1.71)	37.84 (-2.21)
Gold	✓		70.04 (+1.33)	45.9 (+0.96)	35.36 (-1.71)	29.62 (+0.6)	41.47 (+1.42)
Flawed		✓	62.63 (-6.07)	41.46 (-3.49)	34.51 (-2.56)	26.64 (-2.38)	37.30 (-2.76)
Gold		✓	**70.13 (+1.42)**	**46.18 (+1.23)**	**39.17 (+2.09)**	**30.14 (+1.12)**	**41.54 (+1.48)**

Gold AMR vs Flawed AMR. Additionally, we selected 70 AMR samples (labeled as "Flawed") with ambiguities or structural flaws from each of the 10 datasets and refined them using a dual-process correction strategy that combined AMRBART-generated results with manual adjustments, producing high-quality AMRs (labeled "Gold"). Results in Table 4 show that AMR quality significantly impacts model performance. Using flawed AMRs led to performance declines for both direct AMR and AMR-NLD representations, with a more pronounced drop for AMR-NLD. This indirectly validates AMR-NLD's ability to enhance LLMs' understanding of AMR structures. In contrast, with high-quality AMRs, AMR-NLD substantially improved model performance, while direct AMR usage showed limited gains. These results demonstrate that combining high-quality AMR-NLD is more effective in helping models comprehend structured information. This effect is particularly pronounced when the quality of the AMR is high, leading to substantial performance gains.

Fine-tuning Larger Model. To validate the robustness of the proposed method, we selected Llama3.1-70B-Instruct and conducted training-dependent experiments, whose details were consistent with those described for the Llama3.1-8B-Instruct model above, in five tasks shown in the Table 5. The SR used in these experiments was AMR, with a 50%-50% ratio.

Table 5: **Performance of SR-LLM(training-dependent) in Llama3.1-70b-Instruct.** The numbers in parentheses represent the performance differences between adding SR and the control group. Our focus is on the performance variations across different models with different prompts.

AMR	PAWS (F1)	Logic (F1)	Pubmed45 (F1)	WMT16 (BLEU)	SPIDER (EM)
		(a) Vanilla			
✓	68.00	47.13	63.95	28.65	33.71
	60.28 (-7.73)	43.08 (-4.04)	48.82 (-15.13)	27.91 (-0.73)	29.20 (-4.51)
		(b) 50% G(AMR) + 50% G(text)			
	74.74	54.57	76.51	33.73	47.06
✓	**84.56 (+9.81)**	**58.96 (+4.39)**	**81.54 (+5.03)**	**37.00 (+3.27)**	**53.84 (+6.78)**

We can see that, after fine-tuning, the model demonstrated improvements on all tasks, with corresponding values turning positive, more than half of which exceeded 5%. These results further validate the effectiveness of Training-Dependent method on larger-scale models.

Experiment on Traceable LLM. Since NLP datasets have been public for years, their role in modern LLM development is unclear. We validate our approach through experiments on OLMo (Groeneveld et al., 2024), where the training and validation data sources are clearly documented, in five tasks shown in the Table 6. The results show that when the prompt includes SR, its performance is lower than when SR is not included. However, when SR is transformed into SR-NLD, the performance improves signif-

Table 6: **Performance in OLMo.**

SR	SR_NLD (Ours)	PAWS (F1)	Logic (F1)	Pubmed45 (F1)	WMT16 (BLEU)	SPIDER (EM)
		61.52	22.96	60.27	9.67	20.44
✓		57.73	17.97	57.89	11.23	19.79
	✓	**65.40**	**24.89**	**65.19**	**12.44**	**22.20**

icantly. For instance, in the PAWS, the performance increases from 61.52% to 65.40%. This demonstrates the robustness and generalizability of our approach.

5 Related Work

Structure Representations. The SRs, including AMR, PST, and FOL, each unique advantages and applications in specific areas. AMR uses rooted, labeled graphs to abstract syntactic details, offering concise and semantically rich representations (Banarescu et al., 2013). PST, based on Chomsky's generative grammar, employs hierarchical trees to represent sentence syntax and word dependencies (Chomsky, 2014). FOL, as a symbolic logic system, defines objects, their relations, and properties, serving as a key tool in formal logic and reasoning (Enderton, 2001; Barwise, 1977).

Structure Representations Transformation. The SR transformation has long been a critical area of research. Much of the existing work has focused on SR-to-Text approaches, which generate fluent text that aligns with the structure of the SR (Song et al., 2018; Ribeiro et al., 2021; Wang et al., 2020). Meanwhile, a method known as canonical expressions employs rule-based techniques to convert structures into standardized natural language representations, primarily to resolve ambiguities in non-standard sentences (Shin et al., 2021; Roy et al., 2024). Its outputs are essentially normalized texts rather than comprehensive descriptions of the SR's full structure. In contrast, our SR-to-NLD approach preserves the integrity of structured information while enhancing its interpretability through natural language descriptions of the structure.

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. Additionally, Hahn et al. (2022) and (Kalyanpur et al., 2024) advanced formal specification and logical reasoning in LLMs. And An et al. (2024) identified "magic prompts" that improve the performance of NLP tasks by solely focusing on semantic parsing, without the need to provide the actual parsing results. However, Jin et al. (2024) argued that simplely add AMR into prompt 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. 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 in various applications.

7 limitations

Despite SR-NLD's promising performance in certain tasks, its effectiveness remains inconsistent across different LLMs. 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 in various NLP domains.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Kaikai An, Shuzheng Si, Helan Hu, Haozhe Zhao, Yuchi Wang, Qingyan Guo, and Baobao Chang. 2024. Rethinking semantic parsing for large language models: Enhancing llm performance with semantic hints. *arXiv preprint arXiv:2409.14469*.

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

Xuefeng Bai, Yulong Chen, and Yue Zhang. 2022. Graph pre-training for amr parsing and generation. *arXiv preprint arXiv:2203.07836*.

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*.

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

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)*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

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.

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, A. Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Raghavi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, Tushar Khot, William Merrill, Jacob Daniel Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Valentina Pyatkin, Abhilasha Ravichander, Dustin Schwenk, Saurabh Shah, Will Smith, Emma Strubell, Nishant Subramani, Mitchell Wortsman, Pradeep Dasigi, Nathan Lambert, Kyle Richardson, Luke Zettlemoyer, Jesse Dodge, Kyle Lo, Luca Soldaini, Noah A. Smith, and Hanna Hajishirzi. 2024. Olmo: Accelerating the science of language models. *arXiv preprint*.

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*.

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.

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.

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*.

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.

Leonardo FR Ribeiro, Yue Zhang, and Iryna Gurevych. 2021. Structural adapters in pre-trained language models for amr-to-text generation. *arXiv preprint arXiv:2103.09120*.

Subhro Roy, Samuel Thomson, Tongfei Chen, Richard Shin, Adam Pauls, Jason Eisner, and Benjamin Van Durme. 2024. Benchclamp: A benchmark for evaluating language models on syntactic and semantic parsing. *Advances in Neural Information Processing Systems*, 36.

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 Shin, Christopher H Lin, Sam Thomson, Charles Chen, Subhro Roy, Emmanouil Antonios Platanios, Adam Pauls, Dan Klein, Jason Eisner, and Benjamin Van Durme. 2021. Constrained language models yield few-shot semantic parsers. *arXiv preprint arXiv:2104.08768*.

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.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. A graph-to-sequence model for amr-to-text generation. *arXiv preprint arXiv:1805.02473*.

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.

Tianming Wang, Xiaojun Wan, and Hanqi Jin. 2020. Amr-to-text generation with graph transformer. *Transactions of the Association for Computational Linguistics*, 8:19–33.

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.

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*.

A Experimental Details

A.1 Details of Converting SR to SR-NLD

A.1.1 Details of Translating AMR Triplet to Natural Sentence

According to the Figure 6, 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 7.

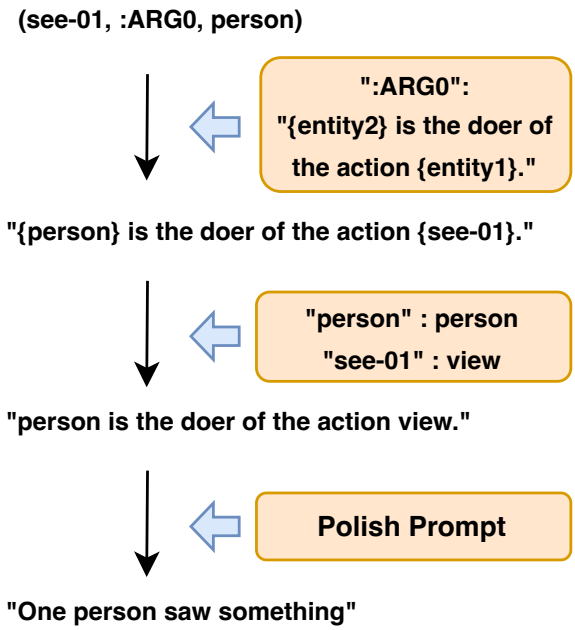


Figure 6: The process of translate entities and relationships into natural language sentences

A.1.2 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 8.

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.

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 7: The prompt of polishing sentence for making AMR-NLD

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 8.

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 9.

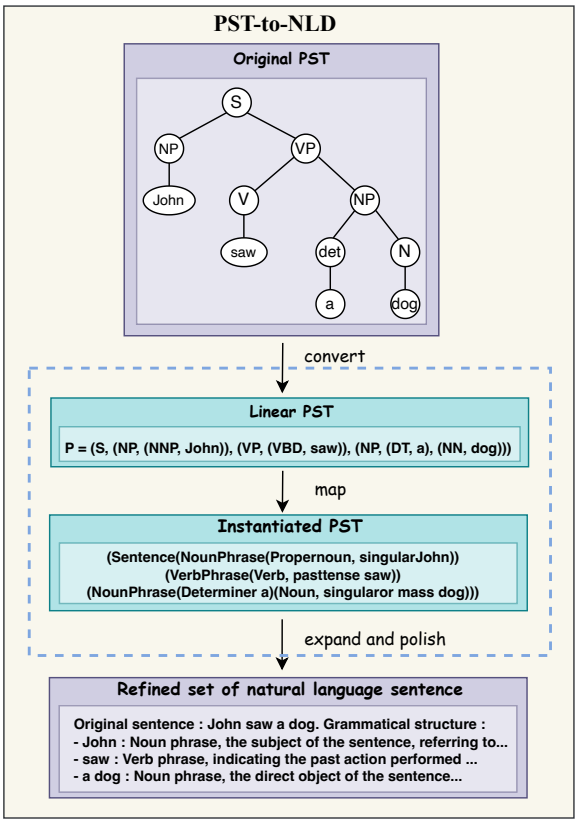


Figure 8: **The Whole process of Making PST-NLD.** The process of creating PST-NLD involves first converting the PST tree structure into a linear sequence of symbols using depth-first search (DFS). Then, a mapping function is applied to translate each node and its children into natural language descriptions. Finally, a language model is used to refine the generated descriptions, making them more natural and coherent.

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 9: The prompt of polishing sentence for making PST-NLD

A.1.3 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 variables, such as $\exists$ (exists) and $\forall$ (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 10.

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.

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 11.

A.2 Complete Fine-tuning Details

We used Meta's Llama-3.1-8B-Instruct as the backbone and conducted fine-tuning on 8 NVIDIA A100-80G GPUs. Optimization was

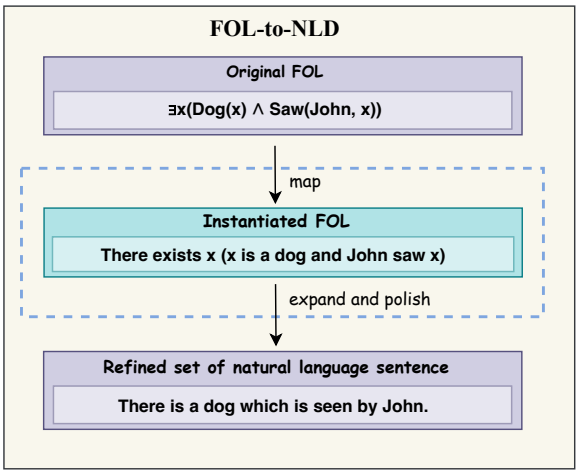


Figure 10: **The Whole process of Making FOL-NLD.** The process of converting FOL to NLD involves first mapping FOL symbols, such as variables, predicates, and logical operators, into natural language using predefined symbol mappings and logical rules. Then, the generated descriptions are refined using a language model to ensure they are coherent and fluent.

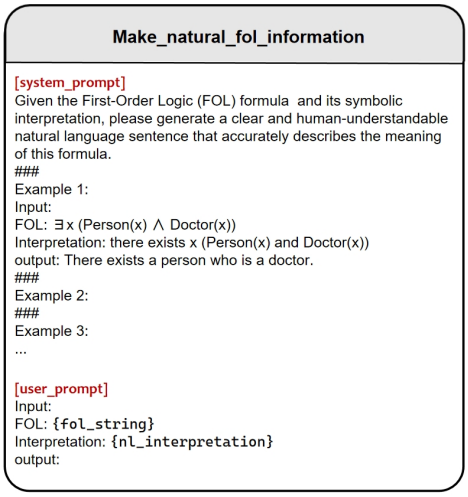


Figure 11: The prompt of polishing sentence for making FOL-NLD

Table 7: 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

performed using the AdamW optimizer with a learning rate of 1e-4 and cosine learning rate decay. The training setup included a per_device_train_batch_size of 16 and gradient_accumulation_steps of 8, yielding an effective global batch size of 1024. A fixed random seed of 42 ensured reproducibility. Each dataset was fine-tuned for 10 epochs, with early stopping to prevent overfitting.

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 7.

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 data were generated using GPT-4o-turbo in a few-shots setting, with the prompt provided in the Figure 12, Figure 13 and Figure 14.

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.

AGNEWS AGNEWS is another large and comprehensive dataset, with a test set of 7,600 examples, which we used directly. Structured

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 12: The prompt of making AMR

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.

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 8.

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

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 13: The prompt of making PST

Table 8: 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

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 15, where a few-shot strategy was employed to guide the model to generate sentences containing different logical fallacies. The generated prompt is

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 14: The prompt of making FOL

shown in Figures 16 and Figures 17. The type of logical error serves as the label, producing complete data points. Structured representations were generated in the same manner as described above.

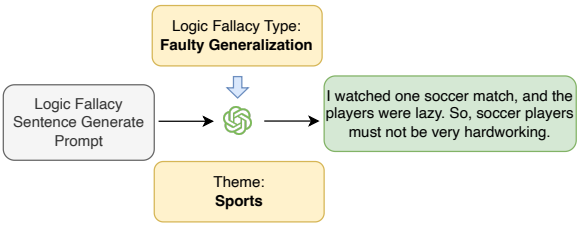


Figure 15: **The synthetic process for LOGIC data.** Taking the "Faulty Generalization" type as an example, we employed a few-shot strategy to guide the model in generating sentences containing the logical fallacy of "Faulty Generalization" To ensure greater sentence diversity, we incorporated a thematic 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.

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

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 16: Logic Fallacy Generate Prompt (a)

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.

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 en-

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:
...

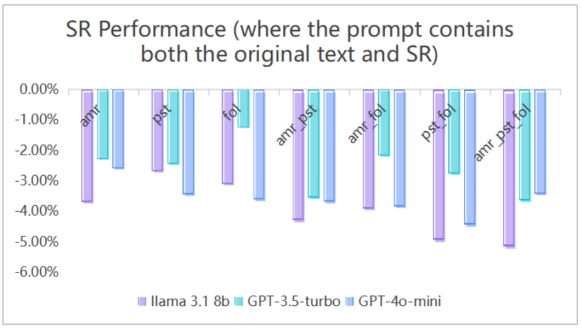
 # fallacy_of_credibility_prompt
You are ... "Fallacy of Credibility" fallacy.
This fallacy occurs when an argument is based on the credibility of the source rather than on logical reasoning or evidence.
###
Example:
...

 # 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:
...

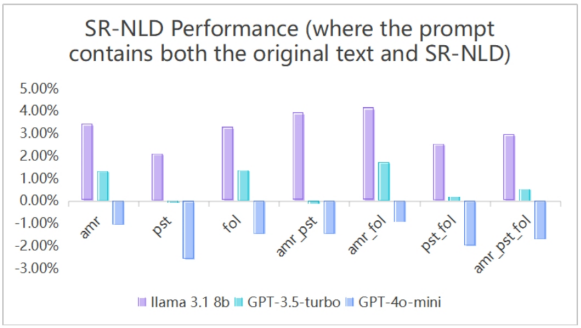
[user_prompt]
input theme: {theme}
output sentence

Figure 17: Logic Fallacy Generate Prompt (b)

hance LLM reasoning capabilities. Figure 18 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.



(a) Average performance of SR Performance in different Tasks



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

Figure 18: **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.

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 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 19. 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 repre-

sents a promising direction for further research.

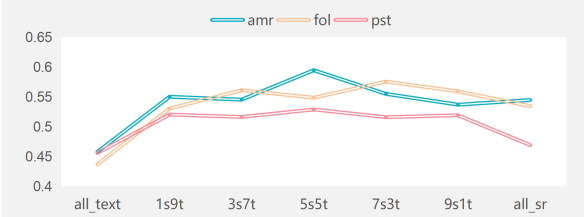


Figure 19: Comparison of average performance of models at different scales in all tasks.

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 9.

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 9: **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 20 shows an example of G(text), Figure 21 shows an example of G(AMR), Figure 22 shows an example of G(PST), and Figure 23 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 20: The Example of G(text)

E Prompt of Testing the SR-LLM

We present the complete prompts for our experiments, including both CoT and One-shot

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 21: The Example of G(AMR)

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 22: The Example of G(PST)

examples, using the SNLI dataset as an illustration in Figures 24, Figures 25 and Figures 26.

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 23: 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 24: 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 25: 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 26: The One-Shot prompt of SNLI's example