Abstract Meaning Representation-Based Logic-Driven Data Augmentation for Logical Reasoning

Anonymous ACL submission

Abstract

Combining large language models with logical reasoning enhances their capacity to address problems in a robust and reliable manner. Nevertheless, the intricate nature of logical reasoning poses challenges to gathering reliable data from the web for building comprehensive training datasets, subsequently affecting the performance on downstream tasks. To address this, we introduce a novel logic-driven data augmentation approach, AMR-LDA. AMR-LDA converts the original text into an Abstract Meaning Representation (AMR) graph, a structured semantic representation that encapsulates the logic structure of the sentence, upon which operations are performed to generate logically modified AMR graphs. The modified AMR graphs are subsequently converted back into text to create augmented data. Notably, our methodology is architecture-agnostic and enhances both generative large language models, such as GPT-3.5 and GPT-4, through prompt augmentation, and discriminative large language models through contrastive learning with logic-driven data augmentation. Empirical evidence underscores the efficacy of our proposed method with improvement in performance across seven downstream tasks, such as reading comprehension requiring logical reasoning, textual entailment, and natural language inference. Furthermore, our method leads on the ReClor leaderboard[1]. The source code and data are publicly available[2].

1 Introduction

Enabling pre-trained large language models (LLMs) to reliably perform logical reasoning is an important step towards strong artificial intelligence (Chollet, 2019). However, data annotation for logical reasoning tasks is a difficult, time-consuming and costly process that has led to the scarcity of large-scale logical reasoning datasets derived from natural language on the web. Therefore, LLMs are usually trained on generic corpora or smaller datasets on logical reasoning that lead to poor generalisation (Wang et al., 2022). The automatic augmentation of logical reasoning data has the potential to enhance the generalisation and performance of LLMs on logical reasoning tasks.

To address this challenge, we propose a logic-driven data augmentation method based on Abstract Meaning Representation (AMR). AMR is a structural representation of the semantic meaning and logical structure of a text via a rooted directed acyclic graph (DAG) (Shou et al., 2022). Figure 1 shows an example of AMR graph. The AMR graph can be easily modified by changing the nodes or arguments to create logically equivalent or nonequivalent graphs. By taking advantage of the ease of logical manipulation of AMR graph and the end-to-end conversion between natural language and AMR graph, our proposed data augmentation is not task-specific or template-dependent, and can generate logically equivalent and nonequivalent sentences that are diverse in language.

In order to improve the performance of LLMs on downstream tasks that require logical reasoning, we investigate two different applications of the proposed logic-driven data augmentation for two different types of language models. In this paper, we call models such as RoBERTa (Liu et al., 2019) and DeBERTa (He et al., 2021) as discriminative large language models, and models like GPT-3.5 (OpenAI, 2023a) as generative LLMs. We improve the reasoning ability of discriminative large language models by applying contrastive learning to identify logically equivalent and nonequivalent sentence pairs generated using the proposed data augmentation, before fine-tuning the model further on downstream tasks. In order to improve the performance of generative LLMs on logical reasoning tasks without fine-tuning, we augment the input

---

[1] https://eval.ai/web/challenges/challenge-page/503/leaderboard/1347

[2] https://bit.ly/3SuiRVi

S1: The girl **believes** that the boy **doesn't work** hard.
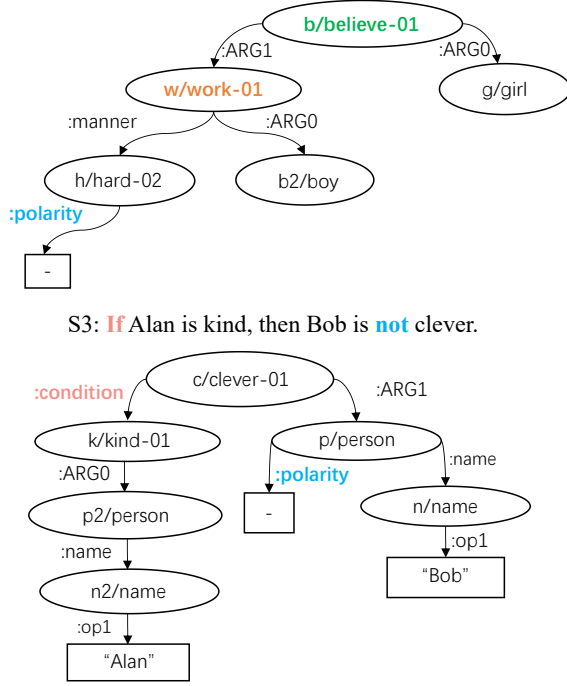S2: The girl **doesn't believe** that the boy **works** hard.

Figure 1: An example of AMR. Two sentences with the same semantic meaning can be represented as the same AMR graph. "b", "g", and "w" are variables. "w/work-01" refers to the variable "w" has an instance relation with the AMR concept "work-01". "work" is the frame from Propbank (Kingsbury and Palmer, 2002) and "-01" is the sense of frame. ":ARG0", ":ARG1", ":condition", ":polarity" are frame arguments, following PropBank instructions. ":condition" and ":polarity -" are used to represent conditional and negative relationships.

prompt by extending the question context and options using data augmentation. We summarize key contributions as follows:

1. We propose an AMR-based logic-driven data augmentation method to automatically construct logically equivalent/nonequivalent sentences.

2. We enhance the logical reasoning of large language models through logical-equivalence-identification contrastive learning and prompt augmentation.

3. The experimental results show that our method can improve large language models' performance on downstream tasks including logical reasoning, textual entailment and natural language inference.

2 Related Work

Logical reasoning is rigorous thinking to derive a conclusion based on a given premise (Seel, 2011; Bronkhorst et al., 2020). Existing datasets on reasoning can be categorised into two levels: sentence level, including tasks like natural language inference that assess if one sentence logically follows from another (e.g., MNLI (Williams et al., 2018), RTE (Wang et al., 2018), MRPC (Dolan and Brockett, 2005), QNLI (Rajpurkar et al., 2016), QQP (Wang et al., 2018)); passage level, which requires logical deduction from given contexts, questions, and multiple choices (e.g., PARARULE (Clark et al., 2021), PARARULE-Plus (Bao et al., 2022)) and reading comprehension tasks (e.g., ReClor (Yu et al., 2020), LogiQA (Liu et al., 2021)). We introduce an abstract meaning representation-based methodology for logic-driven data augmentation aimed at enhancing models' logical reasoning capabilities across these tasks.

There are three primary methods for enhancing the capabilities of pre-trained language models in logical reasoning and general natural language understanding: 1) Data augmentation with fine-tuning, exemplified by AMR-DA (Shou et al., 2022), which employs Abstract Meaning Representation for paraphrasing, and LReasoner (Wang et al., 2022), which uses templates and syntax parsing for constructing logically equivalent sentences; 2) Continual pre-training, with methods like MERIt (Jiao et al., 2022) integrates a meta-path strategy for discerning logical text structures and a counterfactual data augmentation strategy to preclude pre-training shortcuts. IDoL (Xu et al., 2023) utilises six logical indicators (Pi et al., 2022; Prasad et al., 2008) to build a logic pre-training dataset from Wikipedia, enhancing the logical reasoning capabilities of pre-trained models. 3) Prompting, notably Chain-of-Thought prompting (Wei et al., 2022), to improve multi-step logical reasoning performance. Our AMR-LDA surpasses LReasoner-LDA by incorporating a broader range of logical equivalence laws, enabling the automatic construction of more precise logically equivalent sentences. Our contrastive learning method enhance the performance of pre-trained models, including MERIt and IDoL, on logical reasoning tasks. Additionally, Our AMR-based logic-driven prompt augmentation can improve large language models' logical reasoning capabilities, contrasting with the detrimental effects of CoT Prompting and AMR-DA.

2

3 Method

3.1 System Architecture

Our system, shown in Figure 2, features an **AMR-Based Logic-Driven Data Augmentation Module** that parses sentences into AMR graphs, modifies the graphs to generate corresponding logically equivalent and nonequivalent graphs, then converts these back into natural language. The **Logical-Equivalence-Identification Contrastive Learning Module** aims to improve the logical reasoning ability of discriminative large language models by conducting contrastive learning to identify equivalent and nonequivalent sentence pairs, before further fine-tuning the model on downstream tasks. The **Prompt Augmentation Module** is intended to improve the performance of generative autoregressive LLMs on logical reasoning tasks by applying the data augmentation module to the input fed into the models at inference time, without performing any fine-tuning.

3.2 AMR-Based Logic-Driven Data Augmentation

We propose **A**bstract **M**eaning **R**epresentation-based **L**ogic-driven **D**ata **A**ugmentation (**AMR-LDA**) to construct logically equivalent and nonequivalent sentences automatically. For simplicity, we consider only individual sentences, and propositional logic statements expressed in natual language. AMR-LDA involves the following steps: *1)*: Convert a sentence into AMR graph. *2)*: Logically augment the AMR graph. *3)*: Convert the logically augmented AMR graph back into natural language.

Text-To-AMR Parsing A text-to-AMR model is used to parse a sentence into an AMR graph. In this step, the input is a natural language sentence written in English. The output is a rooted, labeled, directed, and acyclic AMR graph that captures the main semantic information of the sentence.

AMR Graph Modification The AMR graph is modified to construct logically equivalent and nonequivalent graphs. To create logically equivalent graphs, we consider four different logical equivalence laws: *double negation*, *commutative*, *implication*, and *contraposition* laws. These laws of logical equivalence are defined below using propositional statements $\mathcal{A}$ and $\mathcal{B}$, followed by examples in natural language (e.g. $\mathcal{A}$ is "Alan is kind" and $\mathcal{B}$ is "Bob is clever").

Logical Equivalence Logical equivalence is a fundamental concept in formal logic (Mendelson, 2009). It can be formally defined as: Two propositions or statement forms P and Q are logically equivalent if they have the same truth value in every possible circumstance, or in every possible model. This can be denoted as $P \equiv Q$. This condition can also be described by the statement: P and Q are logically equivalent if and only if the statement "P if and only if Q" is a tautology. A tautology is a statement that is always true, regardless of the truth values of its components. In terms of truth tables, P and Q are logically equivalent if their truth tables are identical, i.e., P and Q have the same truth value for each possible assignment of truth values to their components.

Definition 1: Contraposition Law

$$(\mathcal{A} \to \mathcal{B}) \Leftrightarrow (\neg\mathcal{B} \to \neg\mathcal{A})$$

If Alan is kind, then Bob is clever. $\Leftrightarrow$ *If Bob is not clever, then Alan is not kind.*

To implement the contraposition law, we first swap the first half of the sentence with the second half if the AMR parser detects that the sentence is a conditional statement (e.g. "if-then", as marked by the blue background in Table 1). In the second step, we construct logically equivalent sentences for the four potential scenarios in which the negation may appear. Here, we use one such scenario as an example. If the first half of the sentence has no negation and the second half of the sentence has no negation either, then we will add the negative polarity argument, ":polarity -", to the first half and the second half of the sentence to construct logically equivalent sentences (marked with the yellow background in Table 1). AMR uses ":polarity -" to represent negation (e.g. "not"). Note that our method is not limited to the word "not", the negative argument ":polarity -" in the AMR graph may represent other negative words in the original sentence. We discuss those cases in Section 3.2 Definition 4 when describing the implementation for double negation law. An example of the augmentation process be found in Figure 8 in Appendices.

Definition 2: Implication Law

$$(\mathcal{A} \to \mathcal{B}) \Leftrightarrow (\neg\mathcal{A} \lor \mathcal{B})$$

If Alan is kind, then Bob is clever. $\Leftrightarrow$ *Alan is not kind or Bob is clever.*
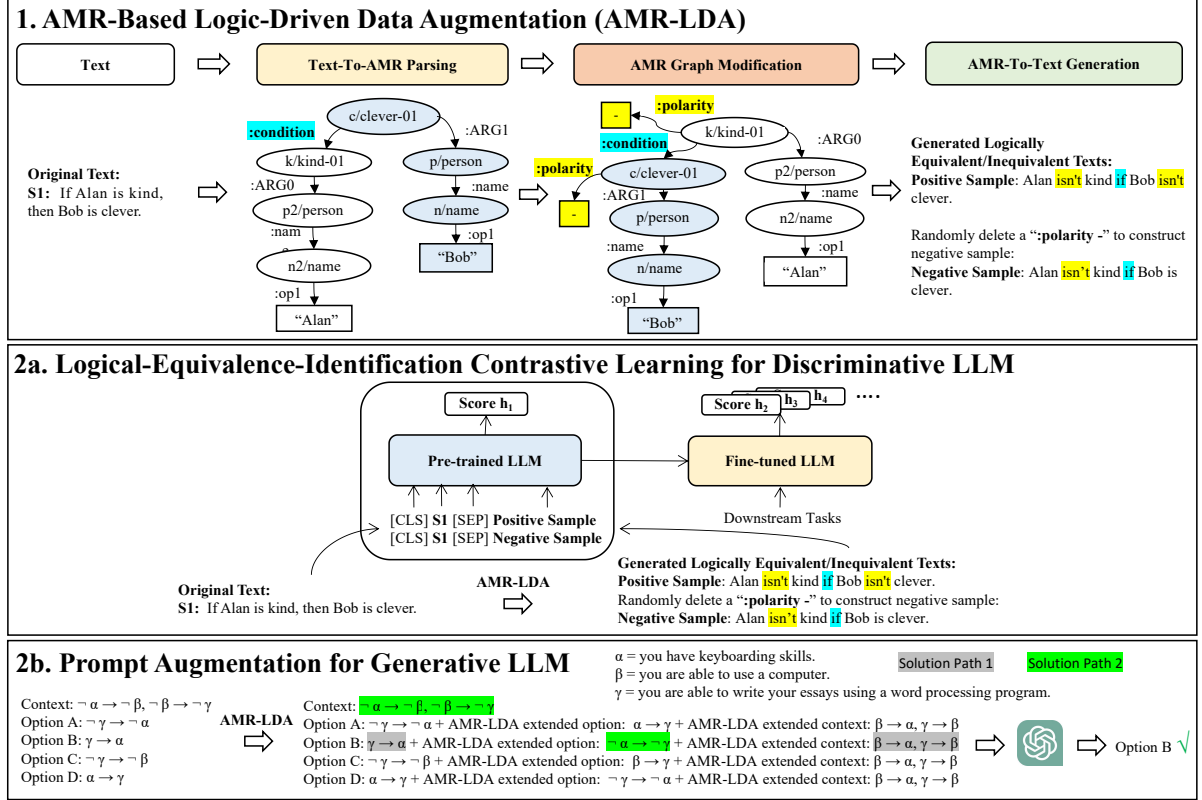
3

Figure 2: Architecture of AMR-LDA (1) and its applications to improve the reasoning performance of discriminative LLMs with contrastive learning (2a) and autoregressive generative LLMs by augmenting input prompts without fine-tuning (2b).

We consider two scenarios. If the sentence is detected by the AMR parser as a conditional statement, then we replace the conditional connective with a disjunction connective (marked with yellow background in Table 1). In the second scenario, if the sentence contains disjunction connectives, we replace the disjunction connective with conditional connective and remove the negative polarity from the AMR graph if it exits. Otherwise, a negative polarity argument will be added. An example can be found in Appendix Figure 6.

Definition 3: Commutative Law

$$(\mathcal{A} \land \mathcal{B}) \Leftrightarrow (\mathcal{B} \land \mathcal{A})$$

Alan is kind and Bob is clever. ⇔ Bob is clever and Alan is kind.

If the AMR graph has a conjunction connective, we swap the order of the first half of the graph with the second half. An example can be found in Table 1 and in Appendix Figure 7. The subsentence "The wolf is fierce" and "the bald eagle is clever" marked as blue have been swapped.

Definition 4: Double Negation Law

$$\mathcal{A} \Leftrightarrow \lnot\lnot\mathcal{A}$$

Alan is kind. ⇔ Alan is not unkind.

We apply the double negation law only to those sentences and their AMR graphs that do not contain the ":polarity -" argument which represents negative polarity. There are several words that can be represented as ":polarity -", such as "not", "no", "never", "none", and "nothing". A representative example can be seen in Table 1 and in Appendix Figure 8. The original sentence is "The bald eagle is strong". The logically equivalent sentence we construct using double negation law is "The bald eagle is not weak", while the logically nonequivalent sentence is "The bald eagle is weak". Note that the generated sentences do not contain the word "not" twice. We avoid generating sentences with "not" appearing multiple times consecutively because they are uncommon and unnatural. The process of applying double negation law is as follows: convert the sentence into an AMR graph;

4

Original sentence	Positive sample	Negative sample
If Alan is kind, then Bob is clever.	Alan **isn't** kind **if** Bob **isn't** clever.	Alan **isn't** kind **if** Bob **is** clever.
	Alan is **not** kind **or** Bob is clever.	Alan is kind **or** Bob is clever.
The bald eagle is strong.	The bald eagle is **not weak**.	The bald eagle is **weak**.
The bald eagle is clever and the wolf is fierce.	**The wolf is fierce** and **the bald eagle is clever**.	**The wolf is** **not** **fierce** and **the bald eagle is** **not** **clever**.

Table 1: Examples of generated logically equivalent (positive) and nonequivalent sentences(negative). The blue background highlights the parts of the original sentence that have been moved from their original positions. The yellow background highlights the change in polarity from the original sentence.

augment the AMR graph by adding a negative polarity argument ": polarity -"; convert the modified AMR graph back into a natural language sentence; lastly, replace the adjective word with its antonym by using WordNet (Miller, 1992). To create logically nonequivalent sentences, we randomly delete or add a negative polarity argument ":polarity -" in the AMR graph. Additionally, we randomly sample a sentence from the corpus and consider it as logically nonequivalent to the original sentence.

AMR-To-Text Generation Lastly, an AMR-to-text model is used to convert the modified AMR graph back into natural language, to generate a sentence that is logically equivalent or nonequivalent to the original sentence.

3.3 Logical-Equivalence-Identification Contrastive Learning

Inspired by SimCSE (Gao et al., 2021) and Sim-CLR (Chen et al., 2020), we propose to improve dicriminative language models' logical reasoning ability by performing contrastive learning to identify logically equivalent and nonequivalent sentence pairs that are generated using AMR-LDA (Figure 2, 2a).

Contrastive Learning The goal of contrastive learning is to minimise the distance of the hidden representations of two similar inputs while maximising the distance between two representations of dissimilar inputs. Our goal is to optimise the model to map logically equivalent sentences to hidden representations that are close to each other.

$$h\left(s, s^{+}\right) \gg h\left(s, s^{-}\right). \quad (1)$$

h is a score function used to measure the distance between two representations. s is an original sentence, s^{+} is a positive sample logically equivalent to the original sentence s, s^{-} is a negative sample logically nonequivalent to the original sentence s.

The expected semantic representation distance between s and s^{+} should be much closer than that of s and s^{-}. The training loss can be written with the following formula:

$$\mathcal{L} = -\sum \log \frac{\exp\left(h\left(+\right)\right)}{\exp\left(h\left(+\right)\right) + \exp\left(h\left(-\right)\right)}, \quad (2)$$

where $h\left(+\right)$ and $h\left(-\right)$ are short for $h\left(s, s^{+}\right)$ and $h\left(s, s^{-}\right)$.

After the contrastive learning step, we further fine-tune the model on downstream tasks, including logical reasoning reading comprehension, natural language inference, and textual entailment.

3.4 Prompt Augmentation

To improve the performance of generative LLMs (e.g., GPT-3.5 or GPT-4) on logical reasoning tasks, we propose augmenting the input prompt using AMR-LDA before feeding it to the model (Figure 2, 2b). In the example from Figure 2, 2b, the context and options are marked in green and grey, respectively. The original Option B is "If you are able to write your essays using a word processing program, then you have keyboarding skills," which cannot be explicitly inferred from the context without using the logical equivalence law (contraposition law). AMR-LDA is able to augment the original option and generate "If you have no keyboarding skills, then you are not able to write your essays using a word processing program," which is logically equivalent to the original Option B, now also marked in green. This augmented Option B can be inferred from the given context. Furthermore, AMR-LDA is also applied to augmenting sentences within the context. The augmented, logically equivalent sentences from the context are "If you are able to use a computer, then you have keyboarding skills. If you are able to write your essays using a word processing program, then you are able to use a computer," which are marked in grey

5

and support the validity of the original Option B. Finally, the augmented option and context are combined and fed as a prompt to GPT-3.5/4. Based on the extended information, we can find two solution paths marked with grey and green backgrounds under **Module 2b** in Figure 2. *Solution Path 1* uses the sentence from the extended context marked with a grey background to support that Option B is correct. *Solution Path 2* uses the sentence from the original context marked with a green background to support that the extended Option B is correct. Consequently, our method provides more solution paths for large language models to more effectively solve complex logical reasoning questions.

4 Experiments

4.1 Datasets

ReClor (Yu et al., 2020) and LogiQA (Liu et al., 2021) are two challenging logical reasoning datasets. ReClor is collected from Graduate Management Admission Test (GMAT) and Law School Admission Test (LSAT). LogiQA is collected by National Civil Service Examination (Liu et al., 2021). Additionally, we performed evaluation on five datasets for natural language inference and textual entailment tasks: MNLI (Williams et al., 2018), RTE (Wang et al., 2018), MRPC (Dolan and Brockett, 2005), QNLI (Rajpurkar et al., 2016), and QQP (Wang et al., 2018). MNLI, RTE, and MRPC assess the relationship between two sentences, while QNLI focuses on the relationship between a question and a sentence, and QQP evaluates the relationship between two questions.

Synthetic Data for Contrastive Learning In this paper, we perform contrastive learning for discriminative large language models on sentences augmented from a synthetic dataset. This dataset contains 14,962 sentences with different combinations of 23 entities, 2 relations and 40 attributes. Synthetic data is used to generate more controllable logical sentences. More details about the synthetic dataset can be found in the Appendix Section E.

4.2 Settings

We conduct all the experiments on 8 NVIDIA A100 GPUs, each with 80G of VRAM. We conducted the primary experiments on the ReClor and LogiQA datasets using three different random seeds, and reported the average values in Table 2. We adopted the parse_xfm_bart_large and T5Wtense models

from AMRLib[3] to perform text-to-AMR and AMR-to-text conversions when generating logically augmented sentence pairs. The reason why we select those two models is explained in subsection C. In our experiments, we use RoBERTa (Liu et al., 2019) and DeBERTa (He et al., 2021) as the discriminative large language models. We also compare our method against MERIt (Jiao et al., 2022) and IDoL (Xu et al., 2023), the leading models on the ReClor leaderboard. As for generative large language models, we utilise GPT-3.5 (gpt-3.5-turbo) (OpenAI, 2023a) and GPT-4 (OpenAI, 2023b). More details about the experiment, case studies and confidence intervals can be found in Appendix Section B, D, D.1, and F.

4.3 Logical-Equivalence-Identification Contrastive Learning for Discriminative LLM

This section evaluates the effectiveness of contrastive learning on synthetic data augmented using AMR-LDA in order to improve the performance of discriminative language models on downstream tasks that require logical reasoning. We compare AMR-LDA against two baseline augmentation methods: AMR-DA (Shou et al., 2022) and LReasoner-LDA (Wang et al., 2022). It is important to note that we do not use the whole system or pipeline from LReasoner, we only use the data augmentation method from LReasoner in our experiment. For each augmentation method, 14,962 pairs of logically equivalent and logically nonequivalent sentences are constructed with a positive to negative sample ratio of 1:1. Twenty percent of the augmented data are used as the validation set during contrastive learning. All the models are further fine-tuned and compared on downstream tasks requiring logical reasoning and natural language inference. The results as shown in Table 2, suggest that the models trained using AMR-LDA perform better in most cases compared with the other augmentation methods.

4.4 Prompt Augmentation for Generative LLM

We adopt GPT-3.5 (gpt-3.5-turbo) (OpenAI, 2023a) and GPT-4 (OpenAI, 2023b) as the generative large language models for evaluating the effectiveness of prompt augmentation using AMR-LDA. The experiments are performed on the ReClor and LogiQA

---

[3]https://amrlib.readthedocs.io/en/latest/models/

6

Models/ Datasets	ReClor				LogiQA		MNLI	MRPC	RTE	QNLI	QQP
	Dev	Test	Test-E	Test-H	Dev	Test			Eval		
RoBERTa	59.73	53.20	72.57	37.97	35.43	34.50	88.95	90.44	83.39	**94.73**	90.89
RoBERTa LReasoner-LDA	59.46	53.66	72.19	39.10	34.81	34.81	89.41	89.46	86.28	94.25	90.01
RoBERTa AMR-DA	58.66	53.93	66.81	**43.80**	36.45	37.22	89.74	90.44	86.28	94.42	92.06
RoBERTa AMR-LDA	**65.26**	**56.86**	**77.34**	40.77	**40.29**	**38.14**	**89.78**	**90.93**	**86.64**	94.49	**93.14**
DeBERTaV2	73.93	70.46	80.82	62.31	39.72	39.62	89.45	89.71	84.48	95.00	**92.54**
DeBERTaV2 LReasoner-LDA	75.73	70.70	84.08	60.17	30.87	28.51	89.23	89.95	87.00	95.15	92.50
DeBERTaV2 AMR-DA	79.06	75.90	84.62	69.04	29.95	30.10	**89.92**	89.71	83.39	95.02	92.42
DeBERTaV2 AMR-LDA	**79.40**	**77.63**	**85.75**	**71.24**	**42.34**	**39.88**	89.67	**90.20**	**88.09**	**95.24**	92.47

Table 2: Comparison between our proposed AMR-LDA and baseline models. We use RoBERTa-Large, DeBERTaV2-XXLarge as the pre-trained models. Our fine-tuned LLMs perform equally well or better than baseline methods.

datasets. The experimental results are shown in Table 3. The models with prompt augmentation achieved better performance in all cases except for the "hard" test set for ReClor. We also compare our method against Chain-of-Thought Prompting (CoT) (Wei et al., 2022) and AMR-DA (Shou et al., 2022) for prompt augmentation. We apply AMR-DA to paraphrase each option and each sentence in the context, and the rest is the same as the AMR-LDA prompt augmentation. We found that CoT and augmentation with AMR-DA caused a decline in performance for both GPT-3.5 and GPT-4 in most cases, except for GPT-4 on LogiQA. The performance drop associated with using CoT Prompting has been reported by (Xu et al., 2023). However, they only sampled 100 cases from the validation set, whereas we use the entire validation set and test set. AMR-DA conducts data augmentation by converting the text into an AMR graph and then randomly selecting one of the following operations: removing, swapping, substituting, or inserting an argument into the graph. The modified AMR graph is then converted back into a new sentence. This modification of the AMR may disrupt the original sentence's structure and introduce noise into the prompt, potentially worsening performance.

GPT-3.5 AMR-LDA performs better than GPT-3.5 on the general test set, which includes both test-E and test-H. The ReClor test set is hidden, so we do not have access to the detailed results for test-E and test-H. Therefore, we cannot provide a clear explanation as to why AMR-LDA seems to decrease the test-H metric for GPT-3.5. However, a detailed examination of the results reveals that GPT-3.5 achieves only a 0.5375 test accuracy on test-H, whereas GPT-4 attains a 0.8857 test accuracy on the same test. Furthermore, GPT-4 with AMR-LDA performs better on all the ReClor and LogiQA

test sets. This suggests that GPT-3.5 might not be as effective in comprehending complex logical reasoning as GPT-4 and GPT-3.5 may struggle with understanding augmented prompts.

Models/Datasets	ReClor				LogiQA	
	Dev	Test	Test-E	Test-H	Dev	Test
GPT-3.5	57.02	56.20	59.31	**53.75**	37.63	37.32
+ CoT	34.80	25.80	27.50	24.46	23.96	24.57
+ AMR-DA	33.20	32.90	34.31	31.78	**40.55**	31.49
+ AMR-LDA	**58.62**	**56.69**	**60.90**	53.39	**40.55**	**39.47**
GPT-4	87.35	89.60	90.90	88.57	43.24	53.88
+ CoT	37.00	24.80	26.13	23.75	23.50	27.03
+ AMR-DA	85.00	85.60	86.36	85.00	51.30	56.06
+ AMR-LDA	**87.73**	**90.20**	**91.59**	**89.11**	51.92	**58.06**

Table 3: Comparison of Chain-of-Thought Prompting (CoT), AMR-DA, and AMR-LDA on GPT-3.5 and GPT-4, and between GPT-3.5 and GPT-4 alone, for evaluation on the ReClor and LogiQA test sets.

Models/Datasets	RoBERTa AMR-LDA	RoBERTa LReasoner-LDA
Depth=1	100.00	100.00
Depth=1 (with altered rules)	**100.00**	99.87
Depth=2	100.00	100.00
Depth=2 (with altered rules)	**99.73**	74.00

Table 4: Comparison between AMR-LDA and LReasoner-LDA with RoBERTa-Large on PARARULE-Plus and PARARULE-Plus (with altered rules). Depth=1 means that only one rule was used to infer the answer. Depth=1 (with altered rules) means one of the rules has been altered using logical equivalence law.

We assessed the robustness of AMR-LDA and LReasoner-LDA models on the PARARULE-Plus dataset (Bao et al., 2022) by modifying the test set with the contraposition law. Examples from this dataset can be found in Appendix Figures 9 and 10. AMR-LDA showed enhanced robustness on these altered tests compared to LReasoner-LDA.

7

Models/Datasets	Con	Con-dou	Con-dou imp	Con-dou imp-com
RoBERTa-Large as backbone model				
ReClor	60.40	60.80	**61.80**	59.80
LogiQA	37.78	33.17	33.94	**38.70**
MNLI	89.55	**90.15**	89.68	89.78
MRPC	90.69	89.22	90.44	**90.93**
RTE	81.23	85.20	84.84	**86.64**
QNLI	94.16	94.05	**94.51**	94.49
QQP	92.12	89.88	92.06	**93.14**
DeBERTaV2-XXLarge as backbone model				
ReClor	**81.80**	72.20	79.40	78.80
LogiQA	32.25	**45.46**	38.24	40.55
DeBERTa-Large as backbone model				
MNLI	**90.80**	90.59	90.68	89.67
MRPC	**90.20**	88.48	89.95	**90.20**
RTE	84.84	87.36	85.56	**88.09**
QNLI	**95.28**	95.04	94.97	95.24
QQP	92.33	92.40	92.29	**92.47**

Table 5: An experiment to assess the influence of different logical equivalence laws on downstream logical reasoning and natural language inference tasks. "Con", "dou", "imp" and "com" are the abbreviation for contraposition law, double negation law, implication law and commutative law. "Con-dou" denotes data constructed using both the contraposition law and the double negation law. Other terms are derived in a similar manner.

Models/Datasets	ReClor				LogiQA	
	Dev	Test	Test-E	Test-H	Dev	Test
DeBERTaV2-XXLarge	73.93	70.46	80.82	62.31	39.72	39.62
+ AMR-LDA-1:1	78.80	76.10	**84.77**	69.28	40.55	41.47
+ AMR-LDA-1:2	80.20	**76.40**	**84.77**	**69.82**	**47.00**	43.93
+ AMR-LDA-1:3	**81.20**	75.70	84.09	69.10	42.70	41.01
DeBERTaV2-XXLarge + MERIt-1:3	80.20	75.80	85.00	68.57	37.32	42.39
+ AMR-LDA-Con-1:3	**82.60**	76.60	86.13	**69.10**	45.00	43.01
+ AMR-LDA-Merged-1:3	81.80	**76.90**	**87.50**	68.57	44.54	**45.62**
DeBERTaV2-XXLarge + IDoL	77.60	74.50	82.95	67.85	39.78	40.24
+ AMR-LDA-Con-1:3	79.20	**77.00**	85.68	**70.17**	**47.61**	**44.54**
+ AMR-LDA-Merged-1:3	**79.40**	75.60	**86.36**	67.14	41.93	41.32

Table 6: An experiment to assess how positive:negative sample ratios affect downstream tasks. AMR-LDA 1:1 means the ratio of positive and negative samples is 1:1.

4.5 Ablation Studies

We perform experiments using a subset of the logical equivalence laws. We present the results in Table 5. This ablation study serves as the basis for our selection of four logical equivalence rules in the main experiment as Table 2 shown. Since the test sets are private and used to rank models on the leaderboard, we evaluated directly using the validation sets instead of the test sets. To make a fair comparison, we ensure the sizes of the training sets are the same for con, con-dou, con-dou-imp and com-dou-imp-com. For this ablation study, we constructed training sets of size 1,000.

We conduct another ablation study where we modify the positive and negative sample ratios. We select DeBERTaV2-XXLarge as the backbone model. We compare the generated data against our AMR-LDA and MERIt. Table 6 shows that a higher proportion of negative samples may help increase the performance on logical reasoning tasks. Furthermore, we chose DeBERTaV2-XXLarge + MERIt-1:3 (Jiao et al., 2022) and DeBERTaV2-XXLarge + IDoL (Xu et al., 2023) as the backbone models. We performed logical equivalence identification contrastive learning, using data constructed solely from the AMR-LDA contraposition law and subsequently merging all four logical equivalence laws. Subsequent fine-tuning on downstream tasks demonstrated that incorporating more logical equivalence laws can enhance the performance of language models on logical reasoning tasks.

5 Conclusion

The sparsity of web data related to logical reasoning constrains the advancement of large language models in their performance on logical reasoning tasks. Existing methods for constructing logically equivalent sentences are restricted to templates and specific datasets. Our AMR-LDA considers more logical equivalence laws than existing methods do, and it does not reply on any ad-hoc templates. We apply AMR-LDA to fine-tuning discriminative LLMs and prompt augmentation of generative LLMs (GPT-3.5 and GPT-4), which yield better results than baseline methods on logical reasoning tasks.

6 Human Evaluation

Human evaluation is conducted to evaluate the correctness and fluency of the logically manipulated sentences generated using AMR-LDA and LReasoner-LDA. We constructed the survey with 20 questions, and each question consists of two randomly selected sentences: one from those generated by our AMR-LDA and the other by LReasoner-LDA, respectively. 45 participants participate to the survey anonymously. We ask them to evaluate the sentences on two aspects: which sentence is logically equivalent to the original sentence or both of them are logically equivalent to the original sentence. The other one is which sentence is more fluent. 63.92% and 76.44% of people prefer AMR-LDA's logically equivalent and fluent sentences over those generated by LReasoner-LDA.

7 Limitations

One limitation of our approach is its reliance on AMR for logic-driven data augmentation, which, while innovative, may not fully capture the intricacies of natural language variation and complex logical constructs encountered in diverse texts. This constraint reflects the broader challenge in NLP of developing models that can understand and reason with the full spectrum of human language, including idiomatic expressions, nuanced context, and varied logical frameworks. Our work makes significant strides in this direction, yet it also highlights the need for continued research to enhance the robustness and adaptability of NLP systems to more closely mirror human-level comprehension and reasoning capabilities.

8 Ethics Statement

All the data used in this paper are either synthetically generated or open-source datasets. All the code used to run the experiments is written using open-source libraries or adapted from published code from other papers. We will also release our code and any synthetically generated data to ensure that the work can be reproduced. The human evaluation has been approved by the Ethics Committee of the main authors' employer.

References

Qiming Bao, Alex Yuxuan Peng, Tim Hartill, Neset Tan, Zhenyun Deng, Michael Witbrock, and Jiamou Liu. 2022. Multi-step deductive reasoning over natural language: An empirical study on out-of-distribution generalisation. In *Proceedings of the 16th International Workshop on Neural-Symbolic Learning and Reasoning as part of the 2nd International Joint Conference on Learning & Reasoning (IJCLR 2022)*, pages 202–217, Cumberland Lodge, Windsor Great Park, United Kingdom.

Hugo Bronkhorst, Gerrit Roorda, Cor Suhre, and Martin Goedhart. 2020. Logical reasoning in formal and everyday reasoning tasks. *International Journal of Science and Mathematics Education*, 18:1673–1694.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

François Chollet. 2019. On the measure of intelligence. *CoRR*, abs/1911.01547.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2021. Transformers as soft reasoners over language. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, IJCAI'20.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: decoding-enhanced bert with disentangled attention. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Fangkai Jiao, Yangyang Guo, Xuemeng Song, and Liqiang Nie. 2022. MERIt: Meta-Path Guided Contrastive Learning for Logical Reasoning. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3496–3509, Dublin, Ireland. Association for Computational Linguistics.

Paul Kingsbury and Martha Palmer. 2002. From Tree-Bank to PropBank. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'02)*, Las Palmas, Canary Islands - Spain. European Language Resources Association (ELRA).

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2021. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, IJCAI'20.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv preprint*, abs/1907.11692.

Elliott Mendelson. 2009. *Introduction to mathematical logic*. CRC press.

George A. Miller. 1992. WordNet: A lexical database for English. In *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992*.

9

OpenAI. 2023a. Chatgpt: Optimizing language models for dialogue.

OpenAI. 2023b. Gpt-4 technical report.

Xinyu Pi, Wanjun Zhong, Yan Gao, Nan Duan, and Jian-Guang Lou. 2022. Logigan: Learning logical reasoning via adversarial pre-training. *Advances in Neural Information Processing Systems*, 35:16290–16304.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind K Joshi, and Bonnie L Webber. 2008. The penn discourse treebank 2.0. In *LREC*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Norbert M Seel. 2011. *Encyclopedia of the Sciences of Learning*. Springer Science & Business Media.

Ziyi Shou, Yuxin Jiang, and Fangzhen Lin. 2022. AMR-DA: Data augmentation by Abstract Meaning Representation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3082–3098, Dublin, Ireland. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Siyuan Wang, Wanjun Zhong, Duyu Tang, Zhongyu Wei, Zhihao Fan, Daxin Jiang, Ming Zhou, and Nan Duan. 2022. Logic-driven context extension and data augmentation for logical reasoning of text. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1619–1629, Dublin, Ireland. Association for Computational Linguistics.

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.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Zihang Xu, Ziqing Yang, Yiming Cui, and Shijin Wang. 2023. IDOL: Indicator-oriented logic pre-training for logical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8099–8111, Toronto, Canada. Association for Computational Linguistics.

Weihao Yu, Zihang Jiang, Yanfei Dong, and Jiashi Feng. 2020. Reclor: A reading comprehension dataset requiring logical reasoning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

A Appendix

B Experiment Setup

We follow the training script from Huggingface and the default hyperparameters[4] to conduct the training and Algorithms 2 and 3 illustrate the negative sample construction and the training process, respectively. For the contrastive learning, we fine-tune RoBERTa-Large, DeBERTa-Large, and DeBERTaV2-XXLarge using the constructed logical equivalence sentence pair from our AMR-LDA, LReasoner's logic-driven data augmentation method (LReasoner-LDA) and AMR-DA data augmentation method. We use DeBERTaV2-XXLarge for ReClor and LogiQA tasks because DeBERTaV2 supports multiple-choice question tasks with a DeBERTaV2ForMultipleChoice head. The hyperparameters for stages 1 and 2 training can be found in Tables 21 and 22.

C Conversion Between Texts and AMR

In order to decide which models to use to perform text and AMR conversions, we experiment with different combinations of text-to-AMR and AMR-to-text models. In the experiment, a sentence is converted to AMR, and then is converted back to text without any modification to the AMR. We pick the combination that can recover the original sentence the most, as measured in BLEU score. The results are reported in Table 7. We find that using parse_xfm_bart large as the AMR parser and T5Wtense as the AMR generator produces the highest BLEU score. Therefore, we se-

---

[4] https://github.com/huggingface/transformers/tree/main/examples/pytorch/text-classification

lect them as the text-to-AMR parser and AMR-to-text generator in all the remaining experiments. Parse_xfm_bart_large is an AMR parser that uses BART-Large as the backbone model (Lewis et al., 2020). T5Wtense is an AMR generator that uses T5 as the backbone model (Raffel et al., 2020).

Text-To-AMR Parser	AMR-To-Text Generator	BLEU
Spring	Spring	25.08
	T5wtense	30.86
	T5	24.76
T5	T5wtense	29.33
	T5	30.82
parse_xfm_bart_large	T5wtense	**38.45**
	T5	30.10

Table 7: Comparison of different combinations of text-to-AMR and AMR-to-text models in recovering original texts after the conversions without any augmentation to the AMR. We adopt the combination with the highest BLEU score in the rest of the experiments.

D Case Studies

We present several case studies comparing our AMR-LDA method with LReasoner-LDA in terms of constructing logically equivalent sentences. These constructions leverage four logical equivalence laws. LReasoner-LDA, however, does not design for the implication law, double negation law, or the commutative law, leading to its inability to handle scenarios that require these laws. Additionally, LReasoner-LDA struggles to construct logically equivalent sentences using the contraposition law when encountering new sentences not found in the ReClor and LogiQA datasets.

Contraposition law	
Original Sentence	If the bald eagle is small, then the mouse is not small.
AMR-LDA	The bald eagle isn't small, unless the mouse is small.
LReasoner-LDA	If it is not small, then it will be not the bald eagle.

Table 8: Logically equivalent sentences constructed by contraposition law.

Contraposition law	
Original Sentence	If the bald eagle is kind, then Dave is not short.
AMR-LDA	If Dave is short, the bald eagle is not kind.
LReasoner-LDA	If it is not kind, then it will be not the bald eagle.

Table 9: Logically equivalent sentences constructed by contraposition law.

Implication law	
Original Sentence	The bear is not sleepy or Bob is not cute.
AMR-LDA	If the bear is not sleepy, then Bob is not cute.
LReasoner-LDA	-

Table 10: Logically equivalent sentences constructed by implication law.

Double negation law	
Original Sentence	The bald eagle is beautiful.
AMR-LDA	The bald eagle isn't ugly.
LReasoner-LDA	-

Table 11: Logically equivalent sentences constructed by double negation law.

Implication law	
Original Sentence	If the lion is not funny, then the tiger is beautiful.
AMR-LDA	Lions aren't funny or Tigers beautiful.
LReasoner-LDA	-

Table 12: Logically equivalent sentences constructed by implication law.

Double negation law	
Original Sentence	The bald eagle is strong.
AMR-LDA	The bald eagle is not weak.
LReasoner-LDA	-

Table 13: Logically equivalent sentences constructed by double negation law.

Commutative law	
Original Sentence	The bald eagle is kind and the wolf is not dull.
AMR-LDA	The wolf is not dull and the bald eagle is kind.
LReasoner-LDA	-

Table 14: Logically equivalent sentences constructed by commutative law.

Commutative law	
Original Sentence	The lion is thin and the dinosaur is not angry.
AMR-LDA	The dinosaur was not angry and the lion was thin.
LReasoner-LDA	-

Table 15: Logically equivalent sentences constructed by commutative law.

D.1 Real World/Long Sentence Case Studies

The appendix of our paper describes Algorithm 1, which uses four lists from Tables 16, 17, 18 and 19 to create synthetic sentences. We've also tested our method on real-world datasets like ReClor and LogiQA that require logical reasoning. Our method, AMR-LDA prompt augmentation, can work with just one list of various sentences. It automatically detects if a sentence can be transformed into a logically equivalent one using a specific logical equivalence law. An example of this application on a real-world sentence is shown in Figure 3. We process sentences from context and options, generating logically equivalent sentences where possible.

Our AMR-LDA can also been applied to long sentences. Our method can generate logically equivalent sentences for long sentences with clear sentence structure using logical equivalence rules (Commutative law) as shown in Figure 4 and 5. The second example shows that our AMR-LDA can understand the effect of that clause on yoga stretching, showing the generalisation advantages of AMR as a semantic representation compared to LReasoner-LDA which relies on a constituency parser and template and fails in this case which is out of templates.

E Synthetic Dataset Construction

Here are the entities, relationships, and attributes we used to construct our synthetic dataset. We used the synthetic dataset to conduct the AMR-based logic-driven data augmentation and logical-equivalence-identification contrastive learning. For the subject, we used "the bald eagle", "the tiger", "the bear", "the lion", "the wolf", "the crocodile", "the dinosaur", "the snake", "the leopard", "the cat", "the dog", "the mouse", "the rabbit", "the squirrel", "Anne", "Alan", "Bob", "Charlie", "Dave", "Erin", "Harry", "Gary", and "Fiona". For the relationships, we used "is" and "is not". For the attributes, we used "kind", "quiet", "round", "nice", "smart", "clever", "dull", "rough", "lazy", "slow", "sleepy", "boring", "tired", "reckless", "furry", "small", "cute", "lovely", "beautiful", "funny", "big", "strong", "awful", "fierce", "heavy", "horrible", "powerful", "angry", "tall", "huge", "short", "thin", "little", "tiny", "wealthy", "poor", "dull", "rough", "bad", and "sad".

Here are the entities, relationships, and attributes we used to fine-tune T5-Large. After T5-Large had been fine-tuned, we used the fine-tuned model to generate logical equivalence sentences as the label for the above synthetic sentences and then conducted the logical-equivalence-identification contrastive learning and downstream task. For the subject, based on the above subject name entities, we add "the duck", "the goat", "the goose", "the donkey", "the cow", "James", "Robert", "John", "Michael", "David", "William", "Richard", "Anthony", "Paul", "Andrew". For the attributes, we add "cautious", "careful", "brainy", "bored", "adorable", "aggressive", "anxious", "dizzy", "depressed", "disturbed", and "awful".

The entity names used for the "change name" experiment in Table 20. For the new entity names that we used "the sheep", "the kitten", "the Garfield", "the lion", "the goat", "the bull", "the cow", "the elephant", "the butterfly", "the fish", "Peter", "Bill", "Tom", "Amy", "Charles", "Tim", "Lucy", and "John".

Table 16, 17, 18, and 19 are the logic pattern and its variation that we consider to replace the original logic pattern for the experiment on Table 20.

To validate whether pre-trained language model can distinguish logically equivalent sentences. We design a preliminary experiment as Table 20 shown. We use RoBERTa-Large to conduct the experiment. We first generate a synthetic test set 1, which includes 1312 test samples with 23 entities, 2 relationships, 40 attributes, and 4 logical equivalence laws (double negation, contraposition, implication, and commutative laws). Model's performance can improve if we fine-tune language model on the logical equivalence training set, which is constructed by our AMR-LDA data augmentation method. Also, The result shows that the model's performance will not drop if we change the entity name or logic pattern, this indicates that the fine-tuned discriminative large language model can handle scenarios requiring greater robustness more effectively.

Here are some synthetic sentence examples and more details for implication, conjunction, disjunction, and negation in the context of AMR-LDA mentioned in Algorithm 1.

Double Negation Law: The original sentence "The bald eagle is strong" is parsed into an AMR graph using a text-to-AMR parser. The parser confirms no negative meanings. To apply the double negation law, negative polarity is added, and an AMR-to-text generator then reforms the sentence. WordNet replaces the adjective with its antonym, creating a logically equivalent sentence.

> **AMR-LDA Prompt Augmentation Case Study**
>
> **GPT-4 Input:** "context": "If you have no keyboarding skills at all, you will not be able to use a computer. And if you are not able to use a computer, you will not be able to write your essays using a word processing program.",
>
> "question": "If the statements above are true, which one of the following must be true?", "answers":
> A. "If you are not able to write your essays using a word processing program, you have no keyboarding skills. *If you have the skill of a keyboard, you can write your essay using a word processing program.If you can use a computer, you have keyboarding skills. If you can write your essay with a word processing program, you can use a computer. Whether you have keyboard skills at all or can't use a computer. Whether you can use a computer or you can't write your own essay with a word processing program.*",
>
> B. "If you are able to write your essays using a word processing program, you have at least some keyboarding skills. *If you don't have at least some keyboard skills, you can't write your essay with a word processing program. If you can use a computer, you have keyboarding skills. If you can write your essay with a word processing program, you can use a computer. Whether you have keyboard skills at all or can't use a computer. Whether you can use a computer or you can't write your own essay with a word processing program.*",
>
> C. "If you are not able to write your essays using a word processing program, you are not able to use a computer. *If you can use a computer, you can write your essay using word processing programs. If you can use a computer, you have keyboarding skills. If you can write your essay with a word processing program, you can use a computer. Whether you have keyboard skills at all or can't use a computer. Whether you can use a computer or you can't write your own essay with a word processing program.*",
>
> D. "If you have some keyboarding skills, you will be able to write your essays using a word processing program. *If you can't write your essay with a word processing program, you don't have some keyboard skills. If you can use a computer, you have keyboarding skills. If you can write your essay with a word processing program, you can use a computer. Whether you have keyboard skills at all or can't use a computer. Whether you can use a computer or you can't write your own essay with a word processing program.*"
>
> **GPT-4 output: B**

Figure 3: Example for using AMR-LDA to augment the prompt from ReClor dataset and their subsequent utilisation as input for GPT-4. Data segments that are marked in bold italics and appear in blue were generated using the contraposition law, while those in brown were generated using the implication law.

Commutative Law: The sentence "The bald eagle is clever and the wolf is fierce" is converted into an AMR graph. The root node "a/and" of this graph, a conjunction argument, allows for the application of the commutative law by swapping arguments. The AMR-to-text generator then produces a new sentence, maintaining logical equivalence.

Implication Law: The sentence "If Alan is kind, then Bob is clever" is parsed into an AMR graph. The method checks for conditional and conclusion arguments. An "or" disjunction replaces the root node, and negative polarity is added to the first half of the sentence. The modified graph is then transformed back into a natural language sentence, ensuring logical equivalence.

Contraposition Law: The same initial sentence "If Alan is kind, then Bob is clever" is analyzed. The contraposition law is applied by swapping the conditional and conclusion arguments in the AMR graph and adding negative modifiers to both. The adjusted graph is then converted back into a logically equivalent sentence.

F Confidence Intervals for the Main Experiments

Here are the confidence intervals for the main experiments in Table 23. We select random seed 0, 21 and 42 to conduct the main experiment on ReClor and LogiQA datasets as shown on Table 23. We utilise a 95% confidence interval to calculate.

> **Long Sentence Example 1:**
>
> **Original sentence:** Sarah woke up early in the morning, and she started her day with a cup of coffee and some light yoga stretches.
>
> **Original sentence's AMR graph:** (a / and :op1 (w / wake-up-02 :ARG1 (p / person :name (n / name :op1 "Sarah")) :time (e / early :op1 (d / date-entity :dayperiod (m / morning)))) :op2 (s / start-01 :ARG0 p :ARG1 (d2 / day :poss p) :ARG2 (a2 / and :op1 (c / coffee :quant (v / volume-quantity :quant 1 :unit (c2 / cup))) :op2 (s2 / stretch-01 :ARG0 p :mod (y / yoga) :ARG1-of (l / light-06) :quant (s3 / some)))))
>
> **Modified AMR graph using AMR-LDA:** (a / and :op1 (s / start-01 :ARG0 p :ARG1 (d2 / day :poss p) :ARG2 (a2 / and :op1 (c / coffee :quant (v / volume-quantity :quant 1 :unit (c2 / cup))) :op2 (s2 / stretch-01 :ARG0 p :mod (y / yoga) :ARG1-of (l / light-06) :quant (s3 / some)))) :op2 (w / wake-up-02 :ARG1 (p / person :name (n / name :op1 "Sarah")) :time (e / early :op1 (d / date-entity :dayperiod (m / morning)))))
>
> **Generated logical equivalence sentence using AMR-LDA:** Sarah started her day with a cup of coffee and some light yoga stretching and woke up early in the morning.

Figure 4: One example uses our AMR-LDA to generate logical equivalence sentences for long sentences. In this case, a logical equivalence sentence is generated using commutative law, and the same color represents the same argument. In this case, the order of the former and latter arguments for the conjunction word "and" has been swapped.

> **Long Sentence Example 2:**
>
> **Original sentence:** Sarah woke up early in the morning, and she started her day with a cup of coffee and some light yoga stretches that will help lose weight.
>
> **Original sentence's AMR graph:** (a / and (a / and :op1 (w / wake-up-02 :ARG1 (p / person :name (n / name :op1 "Sarah")) :time (e / early :op1 (d / date-entity :dayperiod (m / morning)))) :op2 (s / start-01 :ARG0 p :ARG1 (d2 / day :poss p) :ARG2 (a2 / and :op1 (c / coffee :quant (v / volume-quantity :quant 1 :unit (c2 / cup))) :op2 (s2 / stretch-01 :mod (y / yoga) :ARG0-of (h / help-01 :ARG1 (l / lose-01 :ARG1 (w2 / weight))) :ARG1-of (l2 / light-06) :quant (s3 / some)))))
>
> **Modified AMR graph using AMR-LDA:** (a / and :op1 (s / start-01 :ARG0 p :ARG1 (d2 / day :poss p) :ARG2 (a2 / and :op1 (c / coffee :quant (v / volume-quantity :quant 1 :unit (c2 / cup))) :op2 (s2 / stretch-01 :mod (y / yoga) :ARG0-of (h / help-01 :ARG1 (l / lose-01 :ARG1 (w2 / weight))) :ARG1-of (l2 / light-06) :quant (s3 / some)))) :op2 (w / wake-up-02 :ARG1 (p / person :name (n / name :op1 "Sarah")) :time (e / early :op1 (d / date-entity :dayperiod (m / morning)))))
>
> **Generated logical equivalence sentence using AMR-LDA:** Sarah started her day with a cup of coffee and some light yoga stretching to help lose weight, and woke up early in the morning.

Figure 5: One example uses our AMR-LDA to generate logical equivalence sentences for long sentences. In this case, a logical equivalence sentence is generated using commutative law, and the same color represents the same argument. AMR-LDA can understand the effect of that clause on yoga stretching. In this case, the order of the former and latter arguments for the conjunction word "and" has been swapped.

	Logic pattern for double negation law
Original sentence	subject + verb + adj
Positive sample	subject + verb + "not" + the antonym of the adj
Negative sample	subject + verb + "not" + adj

Table 16: We used the logic pattern for double negation law for constructing the test set for the experiment in Table 20.

	Original logic pattern for commutative law	Changed logic pattern
s1	sub1 + verb1 + adj1	sub1 + verb1 + "not" + adj1
s2	sub2 + verb2 + adj2	sub2 + verb2 + "not" + adj2
s3	sub1 + verb1 + "not" + adj1	sub2 + verb2 + "not" + adj2
Original sentence	s1 + "and" + s2	
Positive sample	s2 + "and" + s1	
Negative sample	s1 + "and" + s3	

Table 17: We used the logic pattern for commutative law for constructing the test set for the experiment in Table 20.

	Logic pattern for contraposition law
Original sentence1	"If" + sub1 + verb + adj1 + ", then" + sub2 + verb + adj2
Positive sentence1	"If" + sub2 + verb + "not" + adj2 + ", then" + sub1 + verb + "not" + adj1
Negative sentence1	"If" + sub1 + verb + adj1 + ", then" + sub2 + verb + "not" + adj2
Original sentence2	"If" + sub1 + verb + adj1 + ", then" + sub2 + verb + "not" + adj2
Positive sentence2	"If" + sub2 + verb + adj2 + ", then" + sub1 + verb + "not" + adj1
Negative sentence2	"If" + sub1 + verb + adj1 + ", then" + sub2 + verb + adj2
Original sentence3	"If" + sub1 + verb + "not" + adj1 + ", then" + sub2 + verb + adj2
Positive sentence3	"If" + sub2 + verb + "not" + adj2 + ", then" + sub1 + verb + adj1
Negative sentence3	"If" + sub1 + verb + "not" + adj1 + ", then" + sub2 + verb + "not" + adj2
Original sentence4	"If" + sub1 + verb + "not" + adj1 + ", then" + sub2 + verb + "not" + adj2
Positive sentence4	"If" + sub2 + verb + "not" + adj2 + ", then" + sub1 + verb + "not" + adj1
Negative sentence4	"If" + sub1 + verb + "not" + adj1 + ", then" + sub2 + verb + adj2

Table 18: We used the logic pattern for contraposition law for constructing the test set for the experiment in Table 20.

	Original logic pattern for implication law
Original sentence	"If" + sub1 + verb + adj1 + ", then" + sub2 + verb + adj2
Positive sample	sub1 + verb + "not" + adj1 + "or" + sub2 + verb + adj2
Negative sample	sub1 + verb + "not" + adj1 + "or" + sub2 + verb + "not" + adj2
	Changed logic pattern
Original sentence	sub1 + verb + "not" + adj1 + "or" + sub2 + verb + adj2
Positive sample	"If" + sub1 + verb + adj1 + ", then" + sub2 + verb + adj2
Negative sample	sub1 + verb + "not" + adj1 + "or" + sub2 + verb + "not" + adj2

Table 19: We used the logic pattern for implication law for constructing the test set for the experiment in Table 20.

Test sets ↓; Models →	RoBERTa	Fine-tuned RoBERTa
Test set 1	0.5335	0.8513
Test set 2 (change name)	0.5347	0.8510
Test set 3 (change logic)	0.4672	0.9482

Table 20: Compared fine-tuned RoBERTa-Large and RoBERTa-Large on three different synthetic test sets.

	Stage-1 Fine-tuning	Stage-2 Fine-tuning
Seed	2021	0/21/42
Batch Size	32	16/32
Initial Learning Rate	2e-5	2e-5/3e-6
Learning Rate Scheduler Type	Linear	
Epoch	10	
Num Warmup Steps	0	
Weight Decay	0	
Max Sequence Length	256	
Gradient Accumulation Steps	1	

Table 21: Hyperparameter details for stage-1 fine-tuning and stage-2 fine-tuning except ReClor and LogiQA. Stage-1 fine-tuning means logical-equivalence-identification contrastive learning, and stage-2 fine-tuning means fine-tuning on the downstream tasks.

	Stage-2 Fine-tuning	
	ReClor	LogiQA
Seed	42	
Batch Size	2/4	
Gradient Accumulation Steps	2	
Initial Learning Rate	1e-05/1e-5/3e-6	
Epoch	10	
Adam Betas	(0.9, 0.98)	
Adam Epsilon	1e-6	
No Clip Grad Norm	True	
Warmup Proportion	0.1	
weight_decay	0.01	

Table 22: Model hyperparameter tuning details for stage-2 fine-tuning on ReClor and LogiQA.
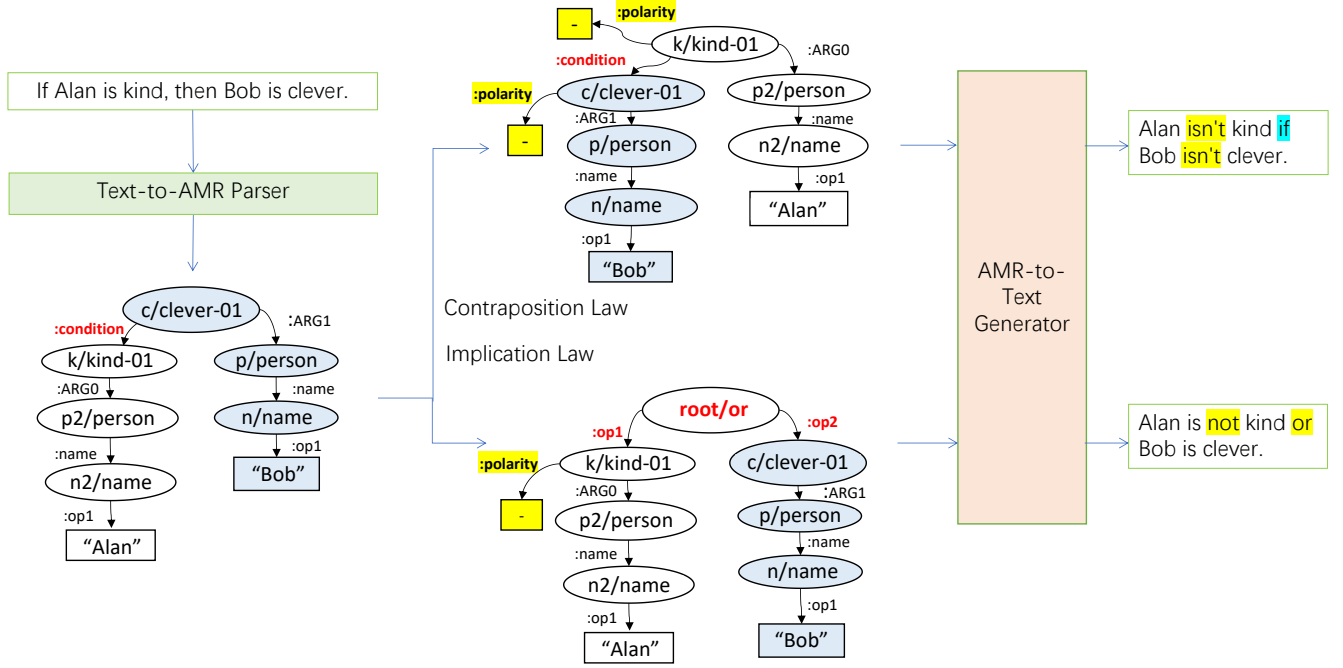
Figure 6: An example of our AMR-based logic-driven data augmentation method using contraposition law and implication law
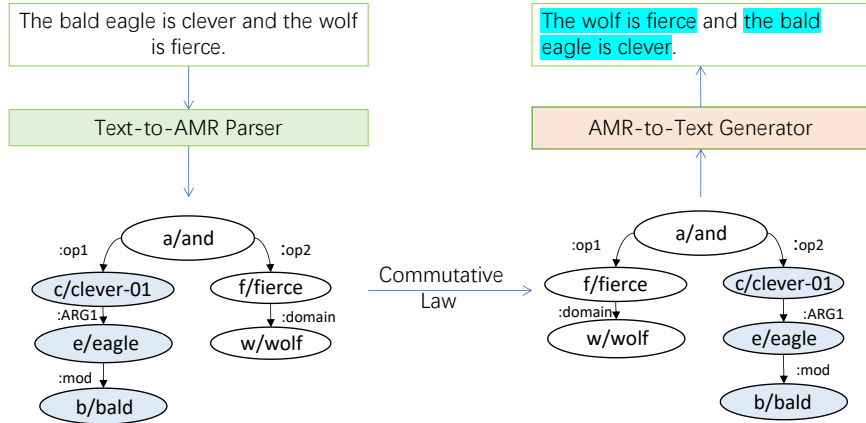

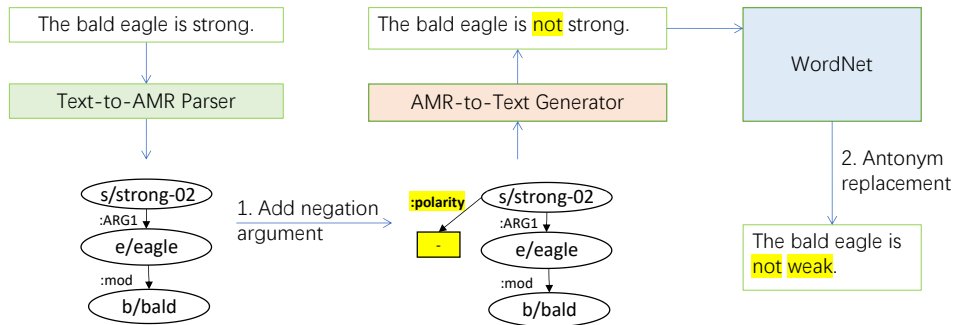Figure 7: An example of our AMR-based logic-driven data augmentation method using commutative law

Figure 8: An example for our AMR-based logic-driven data augmentation method using double negation law

Context (Facts+Rules):

Facts: Alan is tall. Alan is big. Alan is huge. Fiona is thin. Fiona is small. Charlie is quiet. Charlie is smart. Charlie is wealthy. Anne is dull. Anne is sad. Anne is poor.

Rules for Depth=1: If someone is `tall` then they are `quiet`. If someone is `thin` then they are `little`. If someone is `dull and sad` then they are bad. If someone is quiet and smart then they are kind.

Rules for Depth=1 (with altered rules: If someone is `not quiet` then they are `not tall`. If someone is `not little` then they are `not thin`. If someone is `sad and dull` then they are bad. If someone is smart and quiet then they are kind.

Question 1: Alan is quiet? *Label:* True.
Question 2: Alan is not smart? *Label:* False.
Question 3: Fiona is little? *Label:* True.
Question 4: Fiona is not little? *Label:* False.
Question 5: Charlie is kind? *Label:* True.
Question 6: Charlie is not kind? *Label:* False.
Question 7: Anne is bad? *Label:* True.
Question 8: Anne is not bad? *Label:* False.

Figure 9: An example for PARARULE-Plus Depth=1 and Depth=1 (with altered rules). The input includes context (facts + rules) and questions. The output is either true or false. In this example, we use logical equivalence laws (contraposition and commutative laws to extend the sentence in the rule sets to logical equivalence sentences. The highlighted words are the logical equivalence laws that we used. The green and lime green background mean the sentences are constructed by contraposition law, and the cyan background means the sentences are constructed by commutative law.)

Context (Facts+Rules):

Facts: Erin is strong. Erin is tall. Erin is huge. Dave is thin. Dave is short. Fiona is kind. Fiona is wealthy. Fiona is quiet. Bob is sad. Bob is poor. Bob is bad.

Rules for Depth=2: `Strong` people are `kind`. If someone is thin and short then they are little. If someone is sad and poor then they are dull. If someone is `kind and wealthy` then they are `nice`. `All` little people are `small`. All kind people are wealthy. All nice people are smart. `All` dull people are `rough`.

Rules for Depth=2 (with altered rules): If someone is `not kind` then they are `not strong`. If someone is thin and short then they are little. If someone is sad and poor then they are dull. If someone is `not nice` then they are `not both kind and wealthy`. `There are no` little people who are `not small`. All kind people are wealthy. All nice people are smart. `There are no` dull people who are `not rough`.

Question 1: Erin is wealthy? *Label:* True.
Question 2: Erin is not wealthy? *Label:* False.
Question 3: Dave is small? *Label:* True.
Question 4: Dave is not small? *Label:* False.
Question 5: Fiona is smart? *Label:* True.
Question 6: Fiona is not smart? *Label:* False.
Question 7: Bob is rough? *Label:* True.
Question 8: Bob is not rough? *Label:* False.

Figure 10: An example for PARARULE-Plus Depth=2 and Depth=2 (with altered rules). The input includes context (facts + rules) and questions; the output is either "True" or "False". In this example, we use the contraposition law and De Morgan's law to convert sentences in the rule set to logically equivalent sentences. We highlighted the keywords that were changed when the alternative rules were constructed. Green and lime green backgrounds indicate sentences constructed using the contraposition law, while pink and magenta indicate sentences constructed with De Morgan's law.)

Require: Synthetic sentence lists (list1, list2, list3, and list4) generated following the patterns from Table 16, 17, 18, and 19 respectively. total_list = []

  for sent in synthetic_ sentence_lists **do**
    amr_graph = Text-To-AMR-Parser(sent)
    if sent in list1 **then**
      ##double negation law
      if ":polarity -" in amr_graph **then**
        Remove ":polarity -" from the amr_graph
      else
        Add ":polarity -" into the amr_graph
      end if
      aug_text = AMR-To-Text-Generator(amr_graph)
      Use WordNet to replace an adjective word to antonym word from aug_text.
    else if sent in list2 **then**
      ##commutative law
      Switch the order of two arguments.
      aug_text = AMR-To-Text-Generator(amr_graph)
    else if sent in list3 **then**
      ##implication law
      Change the root node as "or".
      if ":polarity -" in a condition argument **then**
        Remove the ":polarity -".
      else
        Add ":polarity -" into the argument.
      end if
      aug_text = AMR-To-Text-Generator(amr_graph)
    else if sent in list4 **then**
      ##contraposition law
      Switch the order of two arguments.
      if ":polarity -" in the argument of the amr_graph **then**
        Remove the ":polarity -".
      else
        Add ":polarity -" into the argument.
      end if
      aug_text = AMR-To-Text-Generator(amr_graph)
    end if
    total_list = total_list.append((sent, aug_text, 1))
  end for
  return total_list

Algorithm 1: AMR-Based Logic-Driven Data Augmentation

Require: Synthetic sentence lists (list1, list2, list3, and list4) generated following the patterns from
Table 16, 17, 18, and 19 respectively. total_list = [], total_list2 = []
for sent in synthetic_ sentence_lists **do**
    amr_graph = Text-To-AMR-Parser(sent)
    if ":polarity -" in amr_graph **then**
      Remove ":polarity -"
    else
      Add ":polarity -" into the amr_graph
    end if
    aug_text = AMR-To-Text-Generator(amr_graph)
    total_list = total_list.append((sent, aug_text, 0))
    for sent in total_list **do**
      random select an index i from total_list
      total_list2 = total_list2.append((sent, total_list[i], 0))
    end for
end for
total_list = total_list.extend(total_list2)
return total_list

Algorithm 2: Negative samples construction

Require: positive_list and negative_list from Algorithm 1 and 2, pre-trained large language model
(LLM),
stage-2 downstream task datasets (ReClor, LogiQA, MNLI, RTE, QNLI, QQP), batch_size bs,
learning_rate lr
Stage-1 fine-tuning
for sents, pos_sents, neg_sents from zip(positive_list, negative_list, bs) **do**
    LLM, Loss = Contrastive_loss(LLM,
    sents, pos_sents, neg_sents, label, lr)
end for
Stage-2 fine-tuning
for sent1, sent2 from zip(downstream_tasks, bs) **do**
    LLM, Loss = Cross_entropy_loss(LLM, sent1, sent2, label, lr)
end for

Algorithm 3: Logical-Equivalence-Identification Contrastive Learning

Model/Datasets	ReClor			
	Dev	Test	Test-E	Test-H
RoBERTa	0.5973 [0.5483,0.6400]	0.5320 [0.5230,0.5400]	0.7257 [0.6950,0.7500]	0.3797 [0.3430,0.4100]
RoBERTa LReasoner-LDA	0.5946 [0.5740,0.6100]	0.5366 [0.5240,0.5400]	0.7219 [0.7040,0.7400]	0.3910 [0.3620,0.4200]
RoBERTa AMR-DA	0.5866 [0.5390,0.6300]	0.5393 [0.5170,0.5600]	0.6681 [0.6420,0.6900]	**0.4380 [0.4170,0.4500]**
RoBERTa AMR-LDA	**0.6526 [0.6050,0.7000]**	**0.5686 [0.5520,0.5800]**	**0.7734 [0.7390,0.8000]**	0.4077 [0.3980,0.4100]
DeBERTaV2	0.7393 [0.6620,0.8100]	0.7046 [0.6080,0.8000]	0.8082 [0.7650,0.8500]	0.6231 [0.4770,0.7700]
DeBERTaV2 LReasoner-LDA	0.7573 [0.6840,0.8300]	0.7070 [0.5950,0.8100]	0.8408 [0.7730,0.9000]	0.6017 [0.4550,0.7400]
DeBERTaV2 AMR-DA	0.7906 [0.7360,0.8400]	0.7590 [0.7370,0.7800]	0.8462 [0.8020,0.8900]	0.6904 [0.6620,0.7100]
DeBERTaV2 AMR-LDA	**0.7940 [0.7760,0.8100]**	**0.7763 [0.7380,0.8100]**	**0.8575 [0.8320,0.8800]**	**0.7124 [0.6640,0.7600]**

Model/Datasets	LogiQA	
	Dev	Test
RoBERTa	0.3543 [0.3060,0.4000]	0.3450 [0.3060,0.3800]
RoBERTa LReasoner-LDA	0.3481 [0.3160,0.3900]	0.3481 [0.3090,0.3800]
RoBERTa AMR-DA	0.3645 [0.2940,0.4400]	0.3722 [0.3450,0.4100]
RoBERTa AMR-LDA	**0.4029 [0.3640,0.4700]**	**0.3814 [0.3450,0.4100]**
DeBERTaV2	0.3972 [0.2270,0.5300]	0.3962 [0.1840,0.5400]
DeBERTaV2 LReasoner-LDA	0.3087 [0.3030,0.3100]	0.2851 [0.2180,0.3600]
DeBERTaV2 AMR-DA	0.2995 [0.2540,0.3600]	0.3010 [0.2730,0.3200]
DeBERTaV2 AMR-LDA	**0.4234 [0.3670,0.4800]**	**0.3988 [0.2570,0.4900]**

Table 23: The confidence intervals for the main experiments conducted on the ReClor and LogiQA datasets. We select random seed 0, 21 and 42 to conduct the main experiment on ReClor and LogiQA datasets. We utilise a 95% confidence interval to calculate the confidence interval.