

End-to-End Argument Mining with LLMs: Representation Insights into Multitask Learning

Zhiqiang Shi

Department of Informatics
King's College London
zhiqiang.shi@kcl.ac.uk

Oana Cocarascu

Department of Informatics
King's College London
oana.cocarascu@kcl.ac.uk

Abstract

Argument mining (AM) traditionally relies on multi-stage pipelines that train separate models for each sub-task, making end-to-end AM challenging and often unstable under multitask learning. In this work, we revisit multitask end-to-end AM in the context of large language models (LLMs) and show that multitask learning behaves fundamentally differently from pre-LLM settings. Across multiple datasets and LLMs, multitask fine-tuning of a single LLM consistently outperforms single-task models, while also exhibiting greater robustness to noisy labels and stronger cross-domain generalization. To understand these qualitatively different behaviors of LLMs, we conduct a layer-wise representation analysis, showing that cross-task features emerge in middle layers, early layers are highly domain-sensitive, and richer middle-layer representations drive performance improvements, robustness, and generalization. We further show that this mechanism extends beyond AM by observing similar patterns across other tasks. Our findings provide new insights into multitask learning with LLMs, highlight its potential for unified, efficient, and robust language understanding, and motivate layer-wise parameter efficient adaptation strategies within and beyond AM.¹

1 Introduction

Argument mining (AM) is an area in natural language processing concerned with extracting argumentative structure from data (Lawrence and Reed, 2019), with key tasks including argument component identification, argument component classification (ACC), argument relation identification (ARI), and argument relation classification (ARC).

End-to-end AM aims to complete the entire AM process in a single step. Prior work has predominantly explored two approaches: multitask learn-

ing, where a single model is jointly trained on multiple AM sub-tasks (Kornblith et al., 2019; Bao et al., 2021; Liu et al., 2023) and task reformulation, in which AM tasks are restructured (Persing and Ng, 2016; Eger et al., 2017; Bao et al., 2022). However, most of these approaches were developed for earlier neural architectures for which naive multitask learning often degrades performance unless combined with architectural constraints, auxiliary objectives, or specialized training curricula (Kuribayashi et al., 2019; Liu et al., 2023).

Whilst LLMs achieve strong performance in AM (Al Zubaer et al., 2023; Cabessa et al., 2025), their effectiveness for end-to-end multitask AM remains largely unexplored. Existing works focus on in-context learning or fine-tuning on a single task, and initial attempts at LLM-based end-to-end AM report performance degradation compared to the single task model (Cabessa et al., 2025). While some works investigate LLM-based multitask learning in computational argumentation (Savigny and Yun, 2025), they do not consider end-to-end AM. Thus, it remains unclear whether the limitations of multitask end-to-end AM stem from the task itself or from constraints of the model architectures.

In this paper, we revisit multitask end-to-end AM in the context of LLMs and demonstrate that multitask fine-tuning of a single LLM consistently improves performance across AM sub-tasks, in contrast to earlier approaches. Our evaluation on multiple datasets and LLMs shows that jointly trained LLMs outperform their single-task counterparts on most sub-tasks, while also exhibiting stronger robustness to label noise, greater stability under task corruption, and better cross-domain generalization.

Further, we investigate why LLMs succeed at multitask AM where prior architectures struggle and conduct a detailed layer-wise representation analysis using multiple metrics. Our analysis reveals that: i) middle layers capture features shared across tasks, enabling effective knowledge trans-

¹Code can be found at: https://github.com/zqshizhiqiang/multitask_argument_mining.

fer; ii) early layers are highly domain-sensitive; iii) richer middle-layer representations are linked to robustness and cross-domain generalization. These results suggest that multitask end-to-end AM benefits from inherent representational properties of LLM architectures. We show that the observed patterns extend beyond AM by replicating our analysis on eight climate-related tasks and four out-of-domain datasets.

Our contributions are as follows: **1)** A systematic investigation of multitask learning for LLM-based end-to-end AM, revealing qualitative differences from pre-LLM behavior; **2)** Comprehensive evaluation across models, datasets, and tasks, demonstrating improved performance, noise robustness, and domain generalization under multitask fine-tuning; **3)** A multi-metric representation analysis, revealing how tasks and domains shape multitask learning in LLM-based end-to-end AM; **4)** Extensions to non-AM tasks, confirming that these representational patterns apply more broadly.

2 Related Work

End-to-end AM. Early AM approaches used supervised learning to train separate models for each task (Niculae et al., 2017; Bouslama et al., 2019) and later pre-trained language models to model contextual information (Srivastava et al., 2022). Recently, LLMs have also been adopted in AM, using in-context learning (Al Zubaer et al., 2023) and supervised fine-tuning (Cabessa et al., 2025).

Different from traditional AM, end-to-end approaches perform the entire process in a single step without the need of separate or hand-craft pipelines. Existing end-to-end AM methods can be broadly categorized as task reformulation and multitask learning. In the former, the AM tasks are reformulated so that the model jointly predicts outputs for all sub-tasks, using methods such as integer linear programming for pipeline-based systems (Persing and Ng, 2016), AM as sequence tagging or dependency parsing (Eger et al., 2017; Ye and Teufel, 2021), generative models with constrained pointer mechanisms (Bao et al., 2022), and LLM-based joint formulations (Cabessa et al., 2025). In multitask learning, the model is fine-tuned on all sub-tasks (Kuribayashi et al., 2019; Bao et al., 2021; Morio et al., 2022; Liu et al., 2023). However, for non-LLM models, fine-tuning jointly across sub-tasks without additional techniques often leads to performance degradation (Kuribayashi et al., 2019;

Liu et al., 2023).² In this work, we show that LLMs exhibit qualitatively different behaviors compared to pre-LLM models.

Representation Analysis Various representation analysis techniques have been used to understand LLMs’ internal representations. Probing approaches (Belinkov, 2022) use classifiers to test whether hidden states encode linguistic or semantic properties. Representation engineering techniques (e.g. RepE (Zou et al., 2025) and adversarial representation engineering (Zhang et al., 2024)) manipulate latent directions to steer model behavior without retraining, while Zhao et al. (2024) use centered kernel alignment (CKA) to analyse the hidden representations of instruction fine-tuned models.

3 Methodology

3.1 LLM based End-to-end Argument Mining with Multitask Learning

We revisit multitask end-to-end AM in the context of LLMs by fine-tuning a single LLM jointly on different AM sub-tasks. We consider three tasks: argument component classification (ACC), argument relation identification (ARI), and argument relation classification (ARC).³ Figure 1 compares AM pipelines using LLMs fine-tuned per task (e.g. as in Cabessa et al. (2025)) with our LLM based-end-to-end multitask learning approach.

3.2 Representation Analysis

To understand why LLMs exhibit qualitatively different behaviors in multitask end-to-end AM compared to pre-LLM models, we employ three techniques to analyze how multitask learning affects LLMs’ internal representations: centered kernel alignment (CKA) to measure layer-wise similarity between multitask and single-task models, maximum mean discrepancy (MMD) to quantify domain invariance, and silhouette score to assess clustering quality across domains. These methods allow us to analyze where cross-task and domain-specific features emerge within the model layers.

²Eger et al. (2017) and Morio et al. (2022) report improvements for multitask learning. However, Eger et al. (2017) do not jointly fine-tune the model on different sub-tasks but reformulate the AM task, with multitask learning introduced through auxiliary tasks. Morio et al. (2022) use a two-stage method, first pre-training the model on all sub-tasks across multiple datasets and then fine-tuning it on the target sub-task.

³See Appendix A.1 for task definitions and Appendix A.2 for the prompts and example data used to fine-tune LLMs.

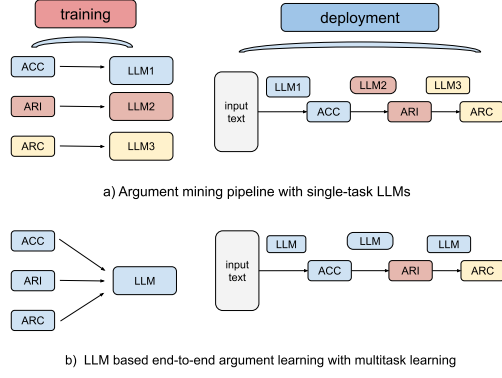


Figure 1: (a) AM pipeline with LLMs per task and (b) LLM-based end-to-end AM with multitask learning, where a single LLM can perform all the sub-tasks.

3.2.1 CKA Similarity

We use CKA (Kornblith et al., 2019) to compute the layer-wise representation similarity between the multitask model and single-task models trained on the same dataset (thus same domain). CKA ranges from 0 to 1, with higher CKA (close to 1) indicating higher similarity. Let $X \in \mathbb{R}^{n \times d_x}$ and $Y \in \mathbb{R}^{n \times d_y}$ denote the representations (for the same n examples) taken from a specific layer of the multitask model and a specific layer of a single-task model, respectively. Following Wang et al. (2024), we utilize the representation of the final token at each layer as the layer-level feature.

The procedure for computing CKA is as follows. We compute the gram matrices using a linear kernel: $K = XX^\top \in \mathbb{R}^{n \times n}$ and $L = YY^\top \in \mathbb{R}^{n \times n}$. We center the gram matrices using the centering matrix $H = I_n - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$ and obtain $K_c = HKH$ and $L_c = HLH$ and apply the Hilbert–Schmidt Independence Criterion $\text{HSIC}(K, L) = \text{tr}(K_c L_c)$. The (linear) CKA between the two layers is the normalized HSIC:

$$\text{CKA}(X, Y) = \frac{\text{HSIC}(K, L)}{\sqrt{\text{HSIC}(K, K)} \sqrt{\text{HSIC}(L, L)}}$$

Suppose we have stored the per-layer embeddings for the two models in tensors $\text{embeddings_A}, \text{embeddings_B} \in \mathbb{R}^{n \times L \times d}$, where n is the number of examples, L is the number of layers, and d is the hidden size. For each pair of layers (i, j) , we form the matrices $X^{(i)} = \text{embeddings_A}[:, i, :] \in \mathbb{R}^{n \times d_x}$ and $Y^{(j)} = \text{embeddings_B}[:, j, :] \in \mathbb{R}^{n \times d_y}$. To quantify the similarity between corresponding layers of the multitask model and the single-task

models, we compute the diagonal elements of the CKA similarity matrix:

$$\text{Sim}(i) = \text{CKA}(X^{(i)}, Y^{(i)}), \quad i = 1, \dots, L$$

where $X^{(i)}$ and $Y^{(i)}$ denote the representations from layer i of the multitask and single-task models, respectively. This yields a vector of layer-wise similarities $\mathbf{s} = [\text{Sim}(1), \text{Sim}(2), \dots, \text{Sim}(L)]$.

3.2.2 MMD and Silhouette Score

We use both maximum mean discrepancy (MMD) (Gretton et al., 2012) and silhouette score (Rousseeuw, 1987) to evaluate the model’s cross domain representations. For the same model, we obtain its hidden states from different domains on the same task and compute the MMD and silhouette score between these domain-specific representations. As with CKA similarity, we use the last token representation as the layer-level representation for each layer.

MMD can be used to measure how close 2 distributions are and is widely used in domain adaptation (Ghifary et al., 2014; Yan et al., 2017) and distribution alignment. Given two sets of embeddings $\mathcal{X} = \{x_i\}_{i=1}^m$ and $\mathcal{Y} = \{y_j\}_{j=1}^n$, the empirical estimate of MMD² with an RBF kernel is:

$$\begin{aligned} \text{MMD}^2(\mathcal{X}, \mathcal{Y}) = & \frac{1}{m(m-1)} \sum_{i \neq i'} k(x_i, x_{i'}) \\ & + \frac{1}{n(n-1)} \sum_{j \neq j'} k(y_j, y_{j'}) \\ & - \frac{2}{mn} \sum_{i,j} k(x_i, y_j) \end{aligned}$$

The Gaussian (RBF) kernel $k(\cdot, \cdot)$ is defined as $k(x, y) = \exp(-\gamma \|x - y\|^2)$ where γ is the kernel bandwidth parameter, often chosen using the median heuristic.

Lower MMD indicates that the two distributions are more similar, which implies better cross-domain generalization.

The silhouette score for a sample i is defined as:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

where $a(i)$ is the average distance from i to all other points in the same cluster (domain), and $b(i)$ is the minimum average distance from i to points in other clusters (domains). The overall silhouette score is the mean over all samples $S = \frac{1}{n} \sum_{i=1}^n s(i)$.

Model	AbstrCT (neo test)		CDCP			PE		
	ACC	ARIC	ACC	ARI	ARC	ACC	ARI	ARC
Best non-LLM	0.928	0.750	0.825	0.678	-	0.892	0.827	0.810
LLaMA3.2 1B S	0.924 $\pm$ 0.006	0.700 $\pm$ 0.009	0.843 $\pm$ 0.005	0.595 $\pm$ 0.003	0.479 $\pm$ 0.000	0.826 $\pm$ 0.010	0.774 $\pm$ 0.006	0.804 $\pm$ 0.010
LLaMA3.2 1B M	0.927 $\pm$ 0.004	0.711 $\pm$ 0.012	0.847 $\pm$ 0.002	0.637 $\pm$ 0.004	0.617 $\pm$ 0.092	0.838 $\pm$ 0.003	0.795 $\pm$ 0.010	0.766 $\pm$ 0.005
LLaMA3.2 3B S	0.934 $\pm$ 0.001	0.681 $\pm$ 0.009	0.853 $\pm$ 0.003	0.672 $\pm$ 0.008	0.479 $\pm$ 0.000	0.867 $\pm$ 0.005	0.756 $\pm$ 0.003	0.876 $\pm$ 0.015
LLaMA3.2 3B M	0.931 $\pm$ 0.002	0.708 $\pm$ 0.008	0.866 $\pm$ 0.004	0.685 $\pm$ 0.015	0.821 $\pm$ 0.177	0.872 $\pm$ 0.009	0.801 $\pm$ 0.015	0.882 $\pm$ 0.021
LLaMA3.1 8B S	0.927 $\pm$ 0.007	0.737 $\pm$ 0.005	0.870 $\pm$ 0.002	0.714 $\pm$ 0.003	0.723 $\pm$ 0.004	0.866 $\pm$ 0.005	0.811 $\pm$ 0.007	0.926 $\pm$ 0.012
LLaMA3.1 8B M	0.933 $\pm$ 0.002	0.751 $\pm$ 0.008	0.872 $\pm$ 0.007	0.700 $\pm$ 0.014	0.761 $\pm$ 0.082	0.882 $\pm$ 0.005	0.824 $\pm$ 0.004	0.899 $\pm$ 0.018
Qwen2.5 1.5B S	0.934 $\pm$ 0.003	0.704 $\pm$ 0.007	0.847 $\pm$ 0.002	0.521 $\pm$ 0.003	0.492 $\pm$ 0.022	0.855 $\pm$ 0.004	0.780 $\pm$ 0.002	0.815 $\pm$ 0.005
Qwen2.5 1.5B M	0.921 $\pm$ 0.005	0.695 $\pm$ 0.008	0.867 $\pm$ 0.003	0.629 $\pm$ 0.020	0.606 $\pm$ 0.021	0.867 $\pm$ 0.004	0.798 $\pm$ 0.009	0.795 $\pm$ 0.014
Qwen2.5 7B S	0.939 $\pm$ 0.004	0.753 $\pm$ 0.004	0.867 $\pm$ 0.002	0.712 $\pm$ 0.009	0.584 $\pm$ 0.000	0.866 $\pm$ 0.003	0.798 $\pm$ 0.004	0.870 $\pm$ 0.016
Qwen2.5 7B M	0.935 $\pm$ 0.003	0.755 $\pm$ 0.004	0.871 $\pm$ 0.020	0.715 $\pm$ 0.017	0.749 $\pm$ 0.095	0.880 $\pm$ 0.006	0.819 $\pm$ 0.006	0.905 $\pm$ 0.005

Table 1: Results on single-task (S) fine-tuning following Cabessa et al. (2025) and multitask (M) fine-tuning the LLM jointly on all sub-tasks. Best non-LLM: Liu et al. (2023) for AbstrCT and PE, Bao et al. (2021) for CDCP.

Because the silhouette score measures how well clusters are separated, a low silhouette score (close to 0) between different domains indicates that the model’s representations across domains are hard to separate. This may suggest that the model has better domain generalization.

4 Experiment Setup

Datasets. We use three datasets from different domains. *PE* (Stab and Gurevych, 2014) contains 402 persuasive essays (1833 paragraphs) annotated with argumentative components and relations. We use paragraph level data without paragraph tags. *CDCP* has 731 user-generated comments from on-line policy discussions. *AbstrCT* (Mayer et al., 2020) is a biomedical dataset of 650 abstracts annotated with argumentative components and relations. We merge MajorClaim and Claim in a single class as in Liu et al. (2023) and use the neo test set as in Morio et al. (2022). As Liu et al. (2023), we consider ACC and ARIC (argument relation identification and classification) for AbstrCT, and ACC, ARI, and ARC for CDCP and PE.

Models. We fine-tune LLaMA 3.1, LLaMA 3.2, and Qwen 2.5 of different sizes, using LoRA (Hu et al., 2022) for parameter efficient fine-tuning. For LLaMA 3.1 8B and Qwen 2.5 7B, we use QLoRA (Dettmers et al., 2023) with 4 bit quantization.⁴ All models are trained on a single NVIDIA A100 GPU with 40 GB VRAM.

Baselines. We include baseline methods (single task fine-tuning) for each model following Cabessa et al. (2025). We also include the best non-LLM results reported in the literature: Liu et al. (2023) for AbstrCT and PE, Bao et al. (2021) for CDCP.

⁴See Appendix A.5 for hyperparameters.

5 Results: Performance, Robustness, and Cross-Domain Generalization

We evaluate whether multitask learning enhances LLM-based end-to-end AM across three dimensions: end-to-end AM performance, robustness to noisy labels, and cross-domain generalization. All results are reported using macro F_1 .⁵

5.1 End-to-end AM performance

Table 1 shows that multitask fine-tuning consistently outperforms single-task fine-tuning on nearly all sub-tasks and datasets. These results contrast with pre-LLM findings, where jointly fine-tuning non-LLM architectures typically degrades performance unless combined by specialized training strategies (Kuribayashi et al., 2019; Bao et al., 2021; Liu et al., 2023). In the LLM setting, however, multitask learning emerges as a simple yet effective approach for unified AM.

5.2 Robustness to Noisy Labels

To evaluate the robustness of multitask models,⁶ we conduct experiments where part of the training data for a specific task is intentionally corrupted. Specifically, we target the PE ACC task by randomly selecting a fraction of its training instances and changing their labels to incorrect classes. These noisy mislabeled examples are then mixed with the remaining original data to form a corrupted ACC dataset, while the data for the other tasks remains unchanged. We use this corrupted ACC data to train (i) a single-task model and (ii) a multitask model alongside the original ARI and ARC data. We focus on ACC as it is a foundational task in

⁵Appendix A.4 details how the macro F_1 score is computed for different tasks.

⁶Here, we refer to robustness against noisy training data.

Corrupt Percentage	Single-task ACC	Multitask		
		ACC	ARI	ARC
Uncorrupted	0.820	0.829	0.808	0.788
0.1	0.793	0.814	0.777	0.750
0.2	0.745	0.768	0.795	0.758
0.3	0.654	0.721	0.794	0.726

Table 2: Effect of corrupting PE ACC task data. Single-task: model trained on corrupted ACC data. Multitask: model trained on mixture of corrupted ACC, and original ARI and ARC data.

the AM pipeline; by corrupting its data, we can assess both model-level robustness to noise and task-level robustness within multitask learning (e.g. how changes in one task affect others).

The results are shown in Table 2 (LLaMA3.2 1B) and Table 11 in Appendix A.6 (Qwen2.5 1.5B). As the corruption rate increases, the performance on the ACC task drops for both models; however, the multitask model outperforms the single-task model and its performance degrades more slowly. This suggests that multitask learning improves robustness to noisy data. For the multitask model, corruption of ACC data also causes a slight drop in other tasks, indicating that shared representations influence cross-task performance. Nevertheless, this drop remains small and stable across corruption levels, demonstrating the task-level robustness of multitask learning. We further discuss the reasons for this robustness in Section 6.2.

5.3 Cross-domain generalization

We analyze domain generalization for both single-task and multitask models.⁷ Since both are trained on the same domain, similar generalization behavior might be expected. To test this, we perform zero-shot transfer and few-shot adaptation. First, the multitask model trained on the PE dataset and the single-task model trained on the PE ACC task are evaluated on ACC test sets from AbstRCT and CDCP. As Table 3 shows, the multitask model surprisingly outperforms the single-task model.

To further test domain adaptation, we conduct experiments using few-shot fine-tuning. We randomly sample 10, 40, and 80 examples from the CDCP ACC task and use them to fine-tune the multitask model trained on the PE dataset, the single-task model trained on PE ACC, and the base

⁷We refer to "domain" as the distribution of texts characterized by its genre, annotation schema, and argumentative conventions. Thus, we treat each dataset as a separate domain.

	Single-task	Multitask
AbstRCT ACC	0.635	0.686
CDCP ACC	0.262	0.372

Table 3: Comparison of cross-domain performance for single-task and multitask models trained on the PE dataset and evaluated on AbstRCT and CDCP. Single-task: model trained on PE ACC task. Multitask: model jointly trained on PE ACC, ARI, and ARC tasks.

Number of examples	From base	Single-task	Multitask
10	0.182	0.322	0.345
40	0.273	0.380	0.409
80	0.484	0.446	0.593

Table 4: Results for using a few examples in a target domain to fine-tune the single-task and multitask models.

LLM.⁸ Table 4 shows that the multitask model consistently outperforms both the single-task and from-base models, indicating that multitask learning is a promising approach for few-shot domain adaptation. We further discuss the reasons for this improved generalization in Section 6.3.

5.4 Summary of Empirical Results

Across all evaluated LLM architectures and datasets, multitask learning reliably improves end-to-end AM performance, enhances robustness to label noise, and substantially strengthens cross-domain generalization. These findings contrast with pre-LLM multitask AM studies, suggesting that LLMs possess internal mechanisms that support stable multitask learning. In Section 6, we analyze these internal mechanisms directly through layer-wise representation analysis.

6 Representation Analysis

6.1 Where do multitask and single-task models diverge? (Layer-wise CKA)

We compute layer-wise CKA (see Section 3.2.1) between the multitask and single-task model trained on the same dataset. The single-task model is fine-tuned on the PE ACC task, while the multitask model is fine-tuned on PE ACC, ARI, and ARC tasks. We evaluate both models on the PE ACC test set, extract their hidden representations, and compute layer-wise similarity using CKA. The same procedure is applied for ARI and ARC tasks.

Figure 2 shows the results for LLaMA 3.2 1B. We observe that early layers of the multitask and

⁸Base LLM refers to instruction-tuned model such as LLaMA 3.2 1B Instruct.

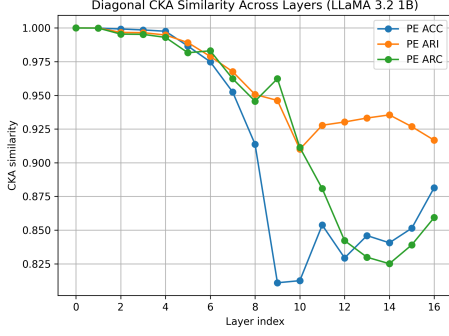


Figure 2: Layer-wise CKA similarity between the multitask model and single-task model for LLaMA 3.2 1B.

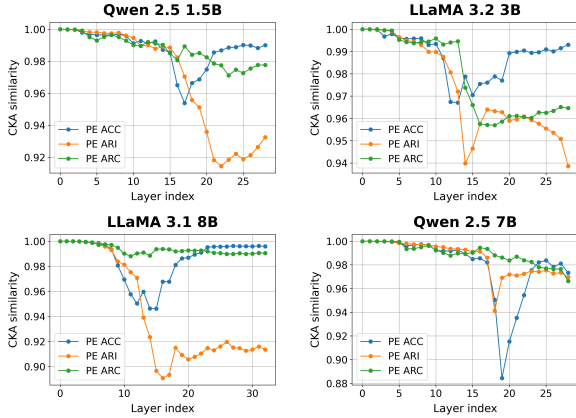


Figure 3: Layer-wise CKA similarity between multitask and single-task models for other models.

single-task models are highly similar, but similarity drops sharply in the middle layers, for example, starting around layer 4 for LLaMA 3.2 1B, and then stabilizes or slightly increases in the final layers. This suggests that multitask models begin encoding cross-task features in the middle layers, while later layers focus more on task-specific representations. These richer intermediate representations help explain the superior performance of multitask models reported in Table 1, as they enhance generalization to unseen examples in the test set. To assess the robustness of these findings, we replicate the experiments on other models and datasets; the results are presented in Figure 3 and Appendix A.6 (Figure 10). Overall, the results show a similar trend, further reinforcing our conclusions.

6.2 Why are multitask models more robust to noisy labels?

As showed in Section 5.2, multitask models exhibit greater robustness to corrupted data compared to single-task models. To investigate what drives these results, we compute layer-wise CKA simi-

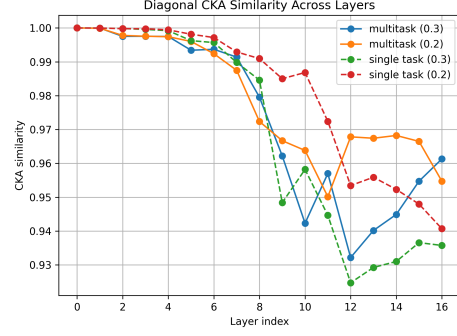


Figure 4: Layer-wise CKA similarity between a model trained on corrupted data across different corrupt percentage of PE ACC and a model trained on clean data.

ilarity between models trained on clean and noisy data. For each multitask or single-task model, we compare the representations of a clean-data model and its corrupted-data counterpart by inferring on the test set and extracting hidden representations.

Figure 4 shows that, for both multitask and single-task models, similarity remains high in the early layers but drops in the middle layers. However, the decrease is slower for the multitask model, and its similarity stabilizes at a higher level beyond layer 12. This trend aligns with Figure 2. Together with the findings in Section 6.1, these results suggest that the richer intermediate representations learned by multitask models contribute to their superior robustness against noisy data.

6.3 How do Multitask LLMs Improve Cross-Domain Generalization?

To analyse why the multitask model has better domain generalization, we use both MMD and silhouette score (see Section 3.2.2) to compare the model’s hidden representations on different domains. Specifically, for each multitask and single-task model trained on the PE dataset, we infer on the PE ACC test set and the CDCP ACC test set to obtain in-domain and out-of-domain representations. We then compute MMD and silhouette scores using these two sets of representations.

Figure 5 shows the results on LLaMA 3.2 1B. We observe that the MMD score increases dramatically in the first two layers, indicating that the model representations on different domains differ significantly in the early layers. This suggests that LLMs tend to encode domain-specific information quickly in their early layers. The score then decreases in the middle layers and remains stable in later layers for both models. Compared to the

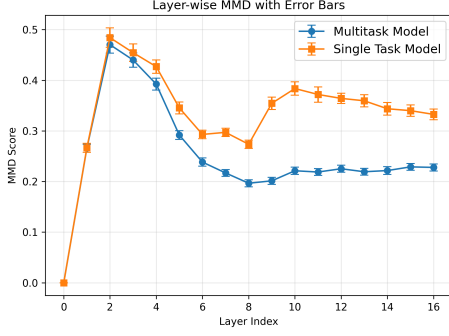


Figure 5: Layer-wise MMD between different domains for multitask and single-task models (LLaMA3.2 1B). The error bars are computed using bootstrap resampling (Efron, 1979).

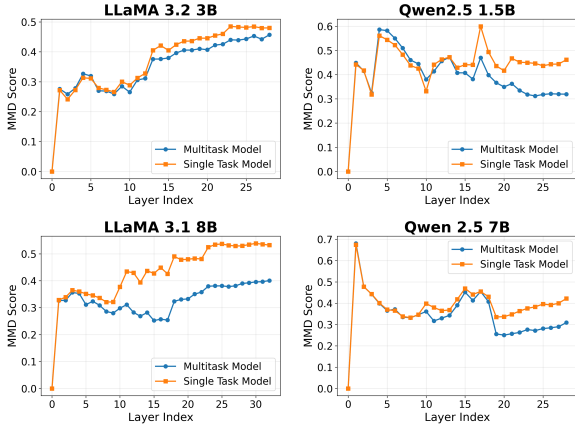


Figure 6: Layer-wise MMD score for other models.

single-task model, the score of the multitask model drops to a lower level, indicating stronger domain invariance. Combined with the findings in Section 6.1 that the multitask model learns richer representations in its middle layers, we infer that the multitask model’s improved domain generalization is due to its richer representations in middle layers.

To ensure the generality of our findings, we replicate these experiments on additional datasets and models (see Figure 6 and Appendix A.6 (Figure 11)). While trends in the middle layers vary across models, a consistent pattern emerges: after the middle layers, the multitask model achieves a lower MMD score, indicating stronger domain invariance. Furthermore, after a certain point (for example, layer 20 for Qwen 2.5 1.5B), the MMD score of multitask and single-task models shows a similar trend. The silhouette score in Appendix A.6 (Figures 12 and 13) shows a similar trend to the MMD score, which further enhances our conclusions.

To better illustrate the difference between multitask and single-task models across domains, we

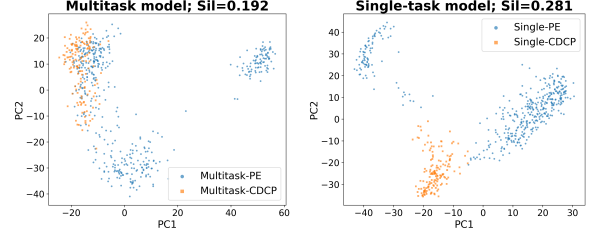


Figure 7: PCA visualization for the representations of multitask and single-task model on different domains. Multitask: multitask model trained on PE ACC, ARI, ARC tasks. Single-task: single-task model trained on PE ACC task. The base model is LLaMA 3.2 1B. Both models are evaluated on PE and CDCP ACC test set.

apply PCA to reduce the dimensionality of the last-layer hidden representations to two and visualize them in Figure 7. Note that the silhouette scores shown are computed in the original representation space. We can see that representations from different domains are less separated for the multitask model than for the single-task model, which further proves the results in Table 3 and Table 4.

6.4 Summary: A layer-wise mechanism for multitask LLMs

Our analyses reveal a consistent mechanism behind the benefits of multitask learning in LLM-based end-to-end AM: i) cross-task features emerge in the middle layers, where multitask models diverge most from single-task models, indicating that these layers encode shared structure across sub-tasks; ii) early layers are highly domain-sensitive, quickly absorbing dataset differences; iii) the richer middle-layer representations learnt through multitask training support greater robustness to noisy labels and stronger cross-domain generalization.

These findings show that multitask learning primarily restructures mid-level representations, enabling stable and domain-invariant features that explain the empirical gains observed in Section 5.

7 Going Beyond Argument Mining

To assess if the representational patterns uncovered in AM reflect a broader property of multitask learning in LLMs, we extend our analysis to eight tasks in the climate domain: Climate Sentiment (Bingler et al., 2023), Climate Stance (Vaid et al., 2022), Environmental Claims (Stammbach et al., 2023), PIRA 2.0 MCQ (Pirozelli et al., 2024), TCFD Recommendations, Climate Commitment, Detection, and Specificity (Bingler et al., 2023). We fine-tune

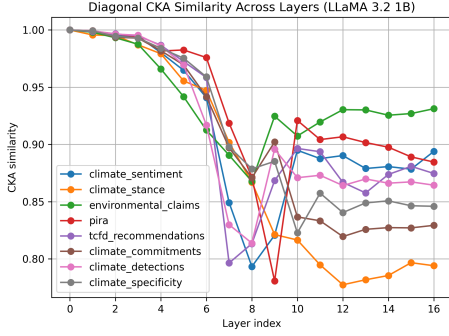


Figure 8: Layer-wise CKA similarity between multitask and single-task model trained on climate domain.

a multitask model on all tasks and train individual models for each task, using LLaMA 3.2 1B as the base model. For cross-domain evaluation, we select four datasets with similar tasks but different domains: IMDB (Maas et al., 2011) (sentiment analysis vs Climate Sentiment), IBM Claim Stance (Bar-Haim et al., 2017) (stance classification vs Climate Stance), CommonsenseQA (Talmor et al., 2019) (multiple-choice QA vs PIRA 2.0 MCQ), and PubMed-200k-RCT (Dernoncourt and Lee, 2017) (paragraph classification vs TCFD Recommendations). We compute layer-wise CKA similarity between the multitask model and each single-task model, with results shown in Figure 8. Additionally, using the same approach as in Section 6.3, we calculate layer-wise MMD scores between in-domain and out-of-domain representations for both multitask and single-task models (Figure 9). The observed trends align with those in Sections 6.1 and 6.3: CKA similarity is high in early layers, drops sharply in the middle layers before it shows a slight increase, and then stabilizes in the final layers. For MMD scores, both models show a sharp increase in early layers, but the multitask model achieves a lower score after the middle layers, indicating stronger domain invariance.

These results further prove our conclusions that the ability of encoding multiple tasks for multitask models happens at middle layers and LLMs tend to encode domain-specific information quickly in their early layers, as well as that the multitask model can have better domain generalization compared to single-task models. When controlling for domain, we do not observe the early-layer similarity drop reported by Zhao et al. (2024). While they attribute this drop to richer representations learned from multiple tasks, our findings in Section 6.3 and Figure 9 suggest a different underlying cause. For

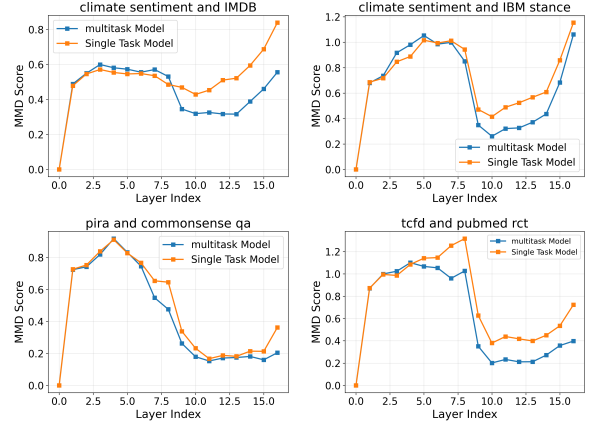


Figure 9: Layer-wise MMD score for multitask and single-task models trained on climate domain.

multitask models trained on multiple domains, the early-layer drop in similarity relative to single-task models is largely due to domain-specific information encoded in these layers. This is supported by Figure 8, where the drop begins after layer 2, compared to layer 4 in Section 6.1 (Figure 2). There, both models shared the same domain as tasks were derived from a single dataset. In contrast, the tasks here, while all climate-related, come from different datasets, introducing domain variation. This accounts for the earlier drop in CKA similarity.

These results suggest that the benefits of multitask LLMs stem from general architectural properties rather than task-specific effects. Multitask training provides more robust and transferable intermediate representations, with strong initialization for downstream tasks and motivating mid-layer focused parameter-efficient adaptation strategies.

8 Conclusions

We presented the first systematic analysis of multitask learning for LLM-based end-to-end AM, our experiments reveal that LLMs show qualitative different behaviors compared to pre-LLM methods for multitask learning based end-to-end AM, where a single multitask fine-tuned LLM outperforms the single task models fine-tuned separately on each task. Moreover, through extensive experiments, we showed that multitask model can obtain better noise robustness and cross domain generalization compared to single-task models, and exhibits task-level robustness. To understand why LLMs behave differently from pre-LLM architectures, we conducted a comprehensive layer-wise representation analysis, revealing that the ability of encoding

multiple tasks for multitask LLMs happens in the middle layers, while early layers reflect more about domain difference. The richer representations in the middle layers of the multitask model contribute to its greater robustness and better cross-domain generalization. Further experiments on eight climate tasks show similar representational patterns, indicating that the benefits of multitask learning arise from general properties of LLM architectures rather than task-specific factors. Future work includes exploring larger models, multilingual extensions, and mid-layer-focused parameter-efficient adaptation strategies.

9 Limitations

While our study provides a comprehensive analysis of multitask learning for LLM-based end-to-end AM, several limitations remain. First, due to computational constraints, we evaluate models up to 8B parameters. It is unclear whether the same layer-wise patterns, particularly the emergence of cross-task features in middle layers, hold for much larger models. Second, we focus on three core AM sub-tasks (ACC, ARI, ARC). Other AM tasks, such as argument quality assessment, stance detection may behave differently under multitask learning. Third, while our representation analyses reveal correlations between layer-wise behavior and empirical gains, establishing stronger causal connections would require controlled interventions or probing methods beyond the scope of this work. Finally, LLMs are trained on broad web-scale corpora, which makes it possible that widely used AM datasets such as PE, CDCP, or AbstrCT may appear in the pretraining data of instruction-tuned LLMs. While this is a general limitation shared across prior LLM-based AM work (e.g., Cabessa et al. (2025)), it does not affect the validity of our comparative findings. In all experiments, both the multitask and single-task models are fine-tuned from the same base LLM, meaning any potential pretraining exposure would influence both models equally. Thus, such exposure cannot explain the performance or representational differences observed between the two training regimes. Nevertheless, we acknowledge this as a broader limitation of LLM research and encourage future work to investigate data-leakage-controlled settings.

References

- Abdullah Al Zubaer, Michael Granitzer, and Jelena Mitrović. 2023. Performance analysis of large language models in the domain of legal argument mining. *Frontiers in artificial intelligence*, 6:1278796.
- Jianzhu Bao, Chuang Fan, Jipeng Wu, Yixue Dang, Jiachen Du, and Ruifeng Xu. 2021. [A neural transition-based model for argumentation mining](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6354–6364, Online. Association for Computational Linguistics.
- Jianzhu Bao, Yuhang He, Yang Sun, Bin Liang, Jiachen Du, Bing Qin, Min Yang, and Ruifeng Xu. 2022. [A generative model for end-to-end argument mining with reconstructed positional encoding and constrained pointer mechanism](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10437–10449, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Roy Bar-Haim, Indrajit Bhattacharya, Francesco Dinuzzo, Amrita Saha, and Noam Slonim. 2017. [Stance classification of context-dependent claims](#). In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 251–261, Valencia, Spain. Association for Computational Linguistics.
- Yonatan Belinkov. 2022. [Probing classifiers: Promises, shortcomings, and advances](#). *Computational Linguistics*, 48(1):207–219.
- Julia Bingler, Mathias Kraus, Markus Leippold, and Nicolas Webersinke. 2023. [How cheap talk in climate disclosures relates to climate initiatives, corporate emissions, and reputation risk](#). Swiss Finance Institute Research Paper No. 22-01.
- Rihab Bouslama, Raouia Ayachi, and Nahla Ben Amor. 2019. Using convolutional neural network in cross-domain argumentation mining framework. In *International Conference on Scalable Uncertainty Management*, pages 355–367. Springer.
- Jérémie Cabessa, Hugo Hernault, and Umer Mushtaq. 2025. [Argument mining with fine-tuned large language models](#). In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 6624–6635, Abu Dhabi, UAE. Association for Computational Linguistics.
- Franck Dernoncourt and Ji Young Lee. 2017. [PubMed 200k RCT: a dataset for sequential sentence classification in medical abstracts](#). In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 308–313, Taipei, Taiwan. Asian Federation of Natural Language Processing.

- Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115.
- Bradley Efron. 1979. [Bootstrap methods: Another look at the jackknife](#). *The Annals of Statistics*, 7(1):1–26.
- Steffen Eger, Johannes Daxenberger, and Iryna Gurevych. 2017. [Neural end-to-end learning for computational argumentation mining](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11–22, Vancouver, Canada. Association for Computational Linguistics.
- Muhammad Ghifary, W Bastiaan Kleijn, and Mengjie Zhang. 2014. Domain adaptive neural networks for object recognition. In *Pacific Rim international conference on artificial intelligence*, pages 898–904. Springer.
- Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. 2012. [A kernel two-sample test](#). *Journal of Machine Learning Research*, 13(25):723–773.
- Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*.
- Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 2019. Similarity of neural network representations revisited. In *International conference on machine learning*, pages 3519–3529. PMIR.
- Tatsuki Kuribayashi, Hiroki Ouchi, Naoya Inoue, Paul Reiser, Toshinori Miyoshi, Jun Suzuki, and Kentaro Inui. 2019. [An empirical study of span representations in argumentation structure parsing](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4691–4698, Florence, Italy. Association for Computational Linguistics.
- John Lawrence and Chris Reed. 2019. [Argument mining: A survey](#). *Computational Linguistics*, 45(4):765–818.
- Boyang Liu, Viktor Schlegel, Riza Batista-Navarro, and Sophia Ananiadou. 2023. [Argument mining as a multi-hop generative machine reading comprehension task](#). In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10846–10858, Singapore. Association for Computational Linguistics.
- Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. [Learning word vectors for sentiment analysis](#). In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.
- Tobias Mayer, Elena Cabrio, and Serena Villata. 2020. Transformer-based argument mining for healthcare applications. In *ECAI 2020 - 24th European Conference on Artificial Intelligence*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 2108–2115. IOS Press.
- Gaku Morio, Hiroaki Ozaki, Terufumi Morishita, and Kohsuke Yanai. 2022. [End-to-end argument mining with cross-corpora multi-task learning](#). *Transactions of the Association for Computational Linguistics*, 10:639–658.
- Vlad Niculae, Joonsuk Park, and Claire Cardie. 2017. [Argument mining with structured SVMs and RNNs](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 985–995, Vancouver, Canada. Association for Computational Linguistics.
- Joonsuk Park and Claire Cardie. 2018. A corpus of erulemaking user comments for measuring evaluability of arguments. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.
- Isaac Persing and Vincent Ng. 2016. [End-to-end argumentation mining in student essays](#). In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1384–1394, San Diego, California. Association for Computational Linguistics.
- Paulo Pirozelli, Marcos M. José, Igor Silveira, Flávio Nakasato, Sarajane M. Peres, Anarosa A. F. Brandão, Anna H. R. Costa, and Fabio G. Cozman. 2024. [Benchmarks for pirá 2.0, a reading comprehension dataset about the ocean, the brazilian coast, and climate change](#). *Data Intelligence*, 6(1):29–63.
- Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65.
- Henri Savigny and Bruno Yun. 2025. Amelia: A family of multi-task end-to-end language models for argumentation. *Argument & Computation*, page 19462174251413169.
- Pranjal Srivastava, Pranav Bhatnagar, and Anurag Goel. 2022. Argument mining using bert and self-attention based embeddings. In *2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, pages 1536–1540. IEEE.
- Christian Stab and Iryna Gurevych. 2014. [Identifying argumentative discourse structures in persuasive essays](#). In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 46–56, Doha, Qatar. Association for Computational Linguistics.

- Dominik Stambach, Nicolas Webersinke, Julia Binger, Mathias Kraus, and Markus Leippold. 2023. [Environmental claim detection](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1051–1066, Toronto, Canada. Association for Computational Linguistics.
- Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. [CommonsenseQA: A question answering challenge targeting commonsense knowledge](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.
- Roopal Vaid, Kartikey Pant, and Manish Shrivastava. 2022. [Towards fine-grained classification of climate change related social media text](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 434–443, Dublin, Ireland. Association for Computational Linguistics.
- Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. [Improving text embeddings with large language models](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11897–11916, Bangkok, Thailand. Association for Computational Linguistics.
- Hongliang Yan, Yukang Ding, Peihua Li, Qilong Wang, Yong Xu, and Wangmeng Zuo. 2017. Mind the class weight bias: Weighted maximum mean discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2272–2281.
- Yuxiao Ye and Simone Teufel. 2021. [End-to-end argument mining as biaffine dependency parsing](#). In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 669–678, Online. Association for Computational Linguistics.
- Yihao Zhang, Zeming Wei, Jun Sun, and Meng Sun. 2024. [Adversarial representation engineering: A general model editing framework for large language models](#). In *Advances in Neural Information Processing Systems*, volume 37, pages 126243–126264. Curran Associates, Inc.
- Zheng Zhao, Yftah Ziser, and Shay B Cohen. 2024. [Layer by layer: Uncovering where multi-task learning happens in instruction-tuned large language models](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15195–15214, Miami, Florida, USA. Association for Computational Linguistics.
- Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xu Wang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, and 2 others. 2025. [Representation engineering: A top-down approach to ai transparency](#). *Preprint*, arXiv:2310.01405.

A Appendix

A.1 Task Definition

ACC: This task involves classifying each argument component (AC) in the input into different types (such as claim or premise), in our work, we ask the model to output the following for the given paragraph:

```
{ "<AC1>": "component_type",
  "<AC2>": "component_type", ... }
```

where "component_type" is replaced by the actual component types.

ARI: This task involves identifying whether different pairs in the given paragraph are related or not. We ask the model to output the following:

```
{ "argument_relations_list":
  [ [ "source AC", "target AC" ], ... ,
    [ "source AC", "target AC" ] ] }
```

"source AC" and "target AC" are argument components (ACs) in the essay text (such as "<AC1>", "<AC2>").

ARC: This task involves classifying the related AC pairs into a specific type, we ask the model to output the following:

```
{ "relation_types": [ relation_type (str)
  , ... , relation_type (str) ] }
```

ARIC: This task involves identifying whether different pairs in the given paragraph are related or not and classifying the related AC pairs into a specific type, we ask the model to output the following:

```
{ "argument_relation_types_list": [ [ "source AC",
  "target AC", "support" | "attack" ], ... , [ "source AC",
  "target AC", "support" | "attack" ] ] }
```

A.2 Prompts and example data

Example prompt for PE ACC task:

You are an expert in Argument Mining. You are given a paragraph containing numbered argument components enclosed in <AC> tags (e.g., <AC1>...</AC1>).

Your task is to classify each argument component as either a "MajorClaim", a "Claim" or a "Premise".

Output Format

Return a single dictionary mapping each <AC> tag to its type:

```
{
  "<AC1>": "MajorClaim" | "Claim" | "Premise",
  "<AC2>": "MajorClaim" | "Claim" | "Premise",
  ...
}
```

Some paragraphs do not contain any argument components (without <AC> tags), output an empty dictionary for these paragraphs.

Do not include any explanations or extra text, just the dictionary.

Example prompt for PE ARI task:

You are an expert in Argument Mining. You are given an essay containing numbered argument components enclosed in <AC> tags (e.g., <AC1>...</AC1>).

Your task is to identify argument relations between argument components in the essay.

Output Format

Return a single json object containing the argument component pairs:

```
{"argument_relations_list": [{"source AC",
"target AC"},..., [{"source AC", "target AC"}]}
"source AC" and "target AC" is argument
components (ACs) in the essay text (such as
"<AC1>", "<AC2>", ...).
```

Some paragraphs do not contain any argument components (without <AC> tags), for these paragraphs, output an empty list for "argument_relations_list".

Do not include any explanations or extra text, just the json object.

Example prompt for PE ARC task:

You are an expert in Argument Mining. You are given an essay containing numbered argument components enclosed in <AC> tags (e.g., <AC1>...</AC1>).

You are also given a list of pairs of related argument components in the form: [{"source AC", "target AC"},..., [{"source AC", "target AC"}]. Your task is to classify each pair of related argument components in the list as either "Support" or "Attack". You must return a list of relation types in following JSON format:

```
{"relation_types": [relation_type (str),
relation_type (str), ..., relation_type (str)]}
```

"source AC" and "target AC" are argument components (ACs) in the essay text (such as "<AC1>", "<AC2>", ...).

Some paragraphs do not contain any argument components (without <AC> tags), for these paragraphs, output an empty list for "relation_types".

Do not include any explanations or extra text, just the json object.

Example prompt for abstrCT dataset:

You are an expert in Argument Mining. You are given a biomedical abstract containing numbered argument components enclosed in <AC> tags (e.g., <AC1>...</AC1>).

Your task is to identify argument relations between argument components in the abstract text and classify their relation type as either "support" or "attack".

Output Format

Return a single json object containing the relation types between different argument components:

```
{"argument_relation_types_list": [{"source AC",
"target AC", "support" | "attack"},..., [{"source
AC", "target AC", "support" | "attack"}]}
"source AC" and "target AC" is argument
components (ACs) in the abstract text (such as
"<AC1>", "<AC2>", ...).
```

Do not include any explanations or extra text, just the json object.

The following examples are some example data for PE dataset. 'instruction' and 'input' will be concatenated as the input to LLMs, while 'output' will be the label.

Example data for PE ACC task:

```
"instruction": "You are an expert in Argument
Mining. You are given a paragraph containing
numbered argument components enclosed in <AC>
tags (e.g., <AC1>...</AC1>).\nYour task is to
classify each argument component as either a
\"MajorClaim\", a \"Claim\" or a
\"Premise\".\nOutput Format\nReturn a single
dictionary mapping each <AC> tag to its
type:\n{\n  \"<AC1>\": \"MajorClaim\" |
\"Claim\" | \"Premise\",
  \"<AC2>\":
\"MajorClaim\" | \"Claim\" | \"Premise\",
  ...
}\nSome paragraphs do not contain any
argument components (without <AC> tags), output
an empty dictionary for these paragraphs.\nDo
not include any explanations or extra text, just
the dictionary.\n"
```

```
"input": "### Here is the paragraph text: On the
other hand , <AC0> the significance of
competition is that how to become more excellence
to gain the victory </AC0> . Hence it is always
said that <AC1> competition makes the society
more effective </AC1> . However , <AC2> when we
consider about the question that how to win the
game , we always find that we need the
cooperation </AC2> . The greater our goal is ,
the more competition we need . <AC3> Take Olympic
games which is a form of competition for instance
, it is hard to imagine how an athlete could win
the game without the training of his or her coach
, and the help of other professional staffs such
as the people who take care of his diet , and
those who are in charge of the medical care
</AC3> . The winner is the athlete but the
success belongs to the whole team . Therefore
<AC4> without the cooperation , there would be no
victory of competition </AC4> ."
```

```
"output": "{ \"<AC0>\": \"Premise\", \"<AC1>\":
\"Claim\", \"<AC2>\": \"Premise\", \"<AC3>\":
\"Premise\", \"<AC4>\": \"Claim\"}"
```

Example data for PE ARI task.

```
"input": "### Here is the paragraph text: However
, some people might ask : what if the criticism
proves to be wrong ? What if that I was at first
right but the seemingly reasonable criticism led
me to the wrong direction ? Doesn ' t insistence
also a good character ? In my point of view ,
<AC0> even if you are right , criticism is still
helpful </AC0> because <AC1> in this case it not
only is a fixer , but it also serves as a
practice to test you how much you know about your
work </AC1> . <AC2> Taking criticism doesn ' t
necessary mean that you have to take it all
</AC2> . <AC3> You should have your own judgments
too </AC3> ! <AC4> And it ' s kind of art to make
the best use of criticism </AC4> .###Here is the
list of pairs of related argument components in
this paragraph: [['<AC1>', '<AC0>'], ['<AC2>',
'<AC0>'], ['<AC3>', '<AC0>'], ['<AC4>',
'<AC0>']]"
```

```
"output": "{ \"relation_types\": [ \"Support\", \"Support\", \"Support\", \"Support\" ] }
```

Example data for abstRCT dataset:

```
"instruction": "You are an expert in Argument Mining. You are given a biomedical abstract containing numbered argument components enclosed in <AC> tags (e.g., <AC1>...</AC1>).\nYour task is to identify argument relations between argument components in the abstract text and classify their relation type as either \"support\" or \"attack\".\n\nOutput\nFormat\nReturn a single json object containing the relation types between different argument components:\n{\n  \"argument_relation_types_list\": [\n    [\"source AC\", \"target AC\", \"support\" | \"attack\"],\n    ...,\n    [\"source AC\", \"target AC\", \"support\" | \"attack\"]\n  ]\n}\n\n\"source AC\" and \"target AC\" is argument components (ACs) in the abstract text (such as \"<AC1>\", \"<AC2>\", ...).\n\nDo not include any explanations or extra text. just the json object.\n"
```

"input": "Here is the abstract text: <AC1> Taxol (paclitaxel; Bristol-Myers Squibb, Wallingford, CT) is a new anticancer agent with activity in a number of human tumors, including epithelial ovarian cancer. </AC1>In nonrandomized trials, doses studied have ranged from 135 mg/m2 to 250 mg/m2 administered over 24 hours with premedication to avoid hypersensitivity reactions (HSRs). This study addressed two questions: the dose-response relationship of Taxol in relapsed ovarian cancer and the safety of a short infusion given with premedication. Women with platinum-pretreated epithelial ovarian cancer and measurable recurrent disease were randomized in a bifactorial design to receive either 175 or 135 mg/m2 of Taxol over either 24 or 3 hours. Major end points were the frequency of significant HSRs and objective response rate. Secondary end points were progression-free and overall survival. Of 407 patients randomized, 391 were eligible and 382 assessable for response. Analysis was performed according to the bifactorial design.<AC2> Severe HSRs were rare (1.5% patients) and were not affected by either dose or schedule. </AC2><AC3> Response was slightly higher at the 175-mg/m2 dose (20%) than at 135 mg/m2 (15%), </AC3><AC4> but this was not statistically significant (P = .2). </AC4><AC5> However, progression-free survival was significantly longer in the high-dose group (19 v 14 weeks; P = .02). </AC5><AC6> Significantly more neutropenia was seen when Taxol was administered as a 24-hour infusion. </AC6><AC7> Response rates were similar in the 24- and 3-hour groups (19% and 16%, respectively; P = .6). </AC7><AC8> No survival differences were noted. </AC8><AC9> The 3-hour infusion of Taxol is safe when given with premedication and is associated with less neutropenia. </AC9>There is a modest dose effect with longer time to progression at 175 mg/m2.<AC10> The observation that longer infusion produces more myelosuppression but does not yield higher response rates should lead to further studies to determine the optimal dose and schedule of this interesting new agent. </AC10>"

"output": "{\n\"argument_relation_types_list\": [[\n\"<AC2>\", \n\"<AC9>\", \n\"support\"], [\n\"<AC3>\", \n\"<AC7>\", \n\"attack\"], [\n\"<AC4>\", \n\"<AC3>\", \n\"attack\"], [\n\"<AC5>\", \n\"<AC9>\", \n\"support\"], [\n\"<AC6>\", \n\"<AC9>\", \n\"support\"], [\n\"<AC7>\", \n\"<AC9>\", \n\"support\"], [\n\"<AC8>\", \n\"<AC9>\", \n\"support\"]]}"

A.3 Dataset licenses, statistics and models

We use three publicly available datasets for AM: AbstRCT (Mayer et al., 2020), CDCP (Park and Cardie, 2018), and Persuasive Essays (PE) (Stab and Gurevych, 2014). All datasets are cited appropriately and distributed for research purposes. Our use aligns with their intended purpose of studying argumentative structures. The datasets are released under the following licenses:

AbstRCT: Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0).

CDCP: ODC Open Database License v1.0 ^{9,10}.

Persuasive Essays: Creative Commons Attribution-NonCommercial-ShareAlike 4.0 (CC BY-NC-SA 4.0).

All datasets are in English.

The train test split for AbstRCT, CDCP and PE dataset are shown in Table 5, Table 6 and Table 7 respectively.

split	num
train	350
gla test	100
mix test	100
neo test	100

Table 5: Train test split for abstRCT dataset.

split	num
train	580
test	150

Table 6: Train test split for CDCP dataset.

split	num
train	1796
test	439

Table 7: Train test split for PE dataset.

We fine-tune and evaluate several open-source large language models using LoRA adapters: LLaMA 3.2: 1B, 3B; LLaMA 3.1: 8B; Qwen 2.5: 1.5B, 7B. All models are used under their respective licenses for non-commercial research. We report parameter sizes, compute resources (A100 GPU), and training configurations in the experimental setup section.

A.4 Explanations for macro F1 score

Following Cabessa et al. (2025), for all tasks (ACC, ARI, ARC, ARIC), the F1 score is computed over the whole test set. Below, we provide explanations for how we compute macro F1.

ACC: for this task, the model will output the following:

```
{ "<AC1>": "component_type",
  "<AC2>": "component_type", ... }
```

We process the data so the ground truth also has the same format. Suppose we obtain the output

⁹<http://www.opendatacommons.org/licenses/odbl/1.0/>

¹⁰<https://spdx.org/licenses/ODbL-1.0.html>

which is a list of dictionaries. Each dictionary corresponds to a paragraph in test set with 2 keys: predict which is the model predictions and label which is the ground truth, both of them are in the above format. Our process of computing F1 score for ACC task is as follows:

1. Initial 2 lists: pred list which will store the components and their labels from model predictions and gt list which will store the components and their labels from ground truth. We concatenate the output from different paragraphs into these 2 lists. Therefore, pred list and gt list are list of dictionaries with predictions or ground truth per article.
2. Loop over all the article, for each article, we have 2 dicts, gt dict and pred dict that contain component and their types for ground truth and model predictions. (1) We convert the keys of gt dict and pred dict to digits (remove < > and AC) to avoid format issues. (2) We then compare whether they have the same keys, if gt dict contains extra keys which means the model misses some components, we add 'none' to the corresponding keys. Note 'none' will not be used as a label when computing F1 score, which means we count the model as wrong for this situation. If pred dict have more keys, we will pop the extra keys. (3) We then append the component types from gt dict and pred dict to y true and y pred in the same order, if a key presented in gt dict but not pred dict, we will append 'none' to y pred.
3. Finally, we compute macro F1 score over y true and y pred with the component types in the datasets (such as 'claim', 'majorclaim') as labels in sklearn.metrics.f1_score.

ARI: For this task, the model will output the following:

```
{"argument_relations_list":
[["source AC", "target AC"],...,
["source AC", "target AC"]]}
```

Suppose we obtain the output which is a list of dictionaries. Each dictionary corresponds to a paragraph in test set with 2 keys: predict which is the model predictions and label which is the ground truth, both of them are in the above format. Our process of computing F1 score for ARI task is as follows:

1. Initialize 2 lists: gt list and pred list which will be used to store whether the given argument component pair has relation or not. These 2 lists contains 2 types of elements: 'relation' which indicates the given argument component pair has relation, 'none' which indicates the given argument component pair does not have relation.
2. Loop over all the paragraph, for each paragraph, we first find the maximum index of argument component in the input max ac. For both model predicted or ground truth 'argument relation list, we use the following method to append 'relation' or 'none' to pred list or gt list: for every possible i, j pair in $0 \leq i, j \leq \max ac$, if [$\langle ACi \rangle$, $\langle ACj \rangle$] appears in model predicted or ground truth 'argument relation list, append 'relation' to pred list or gt list, otherwise, append 'none' to them.
3. Compute macro F1 score over pred list and gt list.

ARC: For this task, the model will output the following:

```
{"relation_types": [relation_type (str)
, ... , relation_type (str)]}
```

Suppose we obtain the output which is a list of dictionaries. Each dictionary corresponds to a paragraph in test set with 2 keys: predict which is the model predictions and label which is the ground truth, both of them are in the above format. Our process of computing F1 score for ARC task is as follows:

1. Initialize 2 lists: gt list and pred list which will be used to store relation types across all the paragraphs for ground truth and model predictions.
2. Loop over all the paragraphs, for each paragraph, let gt types denote the ground truth 'relation types' and pred types denote the model predicted 'relation types'. If the length of gt types is less than pred types, which means the model predicts more than needed, in this case, we pop the extra predictions. If the length of gt types is more than pred types, which means the model predicts less than needed, in this case, we assign 'none' for the lacking predictions. Note

'none' will not be used as a label type when computing F1 score. After these processes, extend gt types and pred types to gt list and pred list.

3. Compute F1 score over gt list and pred list with the relation types in the datasets (such as 'support', 'attack') as labels in sklearn.metrics.f1_score.

ARIC: The computation of F1 score for ARIC is similar to ARI, except that we append the actual relation types (such as 'support', 'attack' to gt list and pred list instead of using 'relation'.

Hyperparameter	value
max_length	1024
per_device_train_batch_size	8
gradient_accumulation_steps	1
num_train_epochs	5
lr_scheduler_type	'cosine'
warmup_ratio	0.1
learning_rate	1e-4
bf16	True

Table 8: Training hyperparameters used in the experiments.

Hyperparameter	value
max_new_token	1024
do_sample	False

Table 9: Generation hyperparameters used in the experiments.

A.5 Experiment configurations

We use HuggingFace SFTTrainer¹¹ for all the experiments. The LoRA config is shown in Table 10. The training hyperparameters are reported in Table 8. We use the hyperparameters in Table 9 for generation. Training a LLaMA 3.2 1B multitask

¹¹https://huggingface.co/docs/trl/v0.26.1/en/sft_trainer#trl.SFTTrainer

Hyperparameter	Value
r(rank)	8
lora_alpha	16
bias	none
target_module	all_linear

Table 10: Hyperparameters for PEFT.

Corrupt Percentage	Single-task ACC	Multitask		
		ACC	ARI	ARC
Uncorrupted	0.855	0.867	0.798	0.795
0.1	0.849	0.854	0.788	0.730
0.2	0.822	0.834	0.803	0.792
0.3	0.779	0.847	0.780	0.778

Table 11: Effect of corrupting PE ACC task data. Single-task: model trained on corrupted ACC data. Multi-task: model trained on mixture of corrupted ACC, and original ARI and ARC data. The base LLM model is Qwen2.5 1.5B.

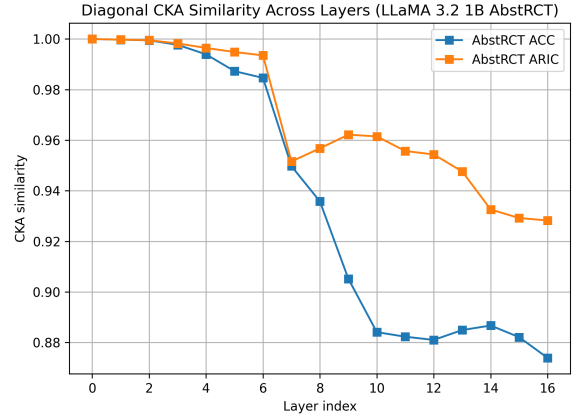


Figure 10: Layer-wise CKA similarity between multitask and single-task model on AbstrCT dataset for LLaMA 3.2 1B.

model on PE ACC, ARI, ARC tasks takes around 30 minutes.

A.6 Additional experiment results

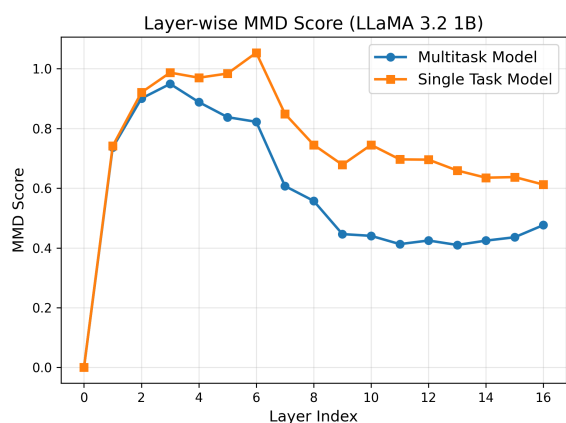


Figure 11: Layer-wise MMD score between PE and AbstrCT dataset for LLaMA 3.2 1B.

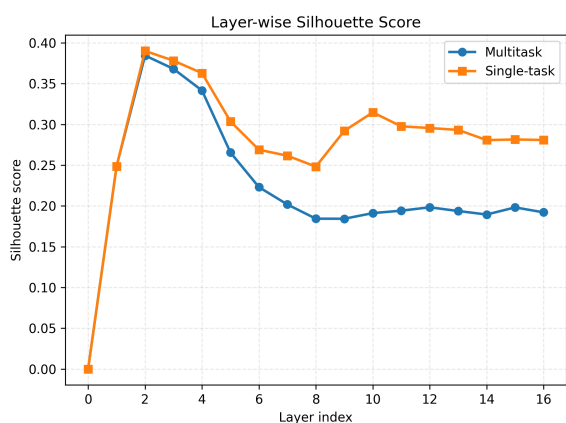


Figure 12: Layer-wise silhouette score between PE and CDCP for LLaMA 3.2 1B.

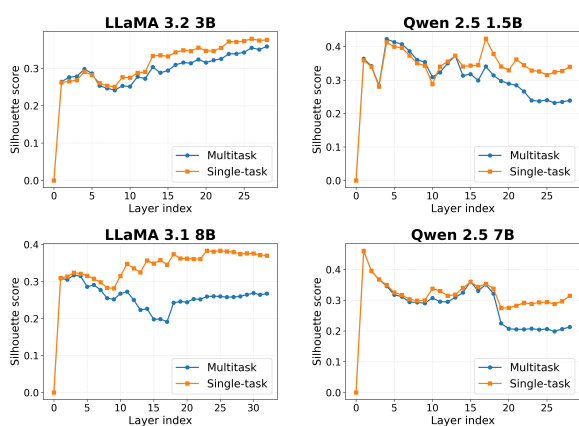


Figure 13: Layer-wise silhouette score for other models.