PITA: Prompting Task Interaction for Argumentation Mining

Yang Sun[1,2*], **Muyi Wang**[1*], **Jianzhu Bao**[1,2], **Bin Liang**[1,4†], **Xiaoyan Zhao**[4]
Caihua Yang[1,5] **Min Yang**[3†] and **Ruifeng Xu**[1,2,5†]
[1] Harbin Institute of Technology, Shenzhen, China [2] Peng Cheng Laboratory, Shenzhen, China
[3] SIAT, Chinese Academy of Sciences, Shenzhen, China [4] The Chinese University of Hong Kong
[5] Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies
yang.sun@stu.hit.edu.cn, 200210231@stu.hit.edu.cn, jianzhubao@gmail.com
bin.liang@cuhk.edu.hk, xzhao@se.cuhk.edu.hk, 21s051020@stu.hit.edu.cn
min.yang@siat.ac.cn, xuruifeng@hit.edu.cn

Abstract

Argumentation mining (AM) aims to detect the arguments and their inherent relations from argumentative textual compositions. Prior methods are afflicted by a sequential feature decoding paradigm, wherein they initially address the features of argumentation components (ACs) for argumentative relation type classification (ACTC) subtask. Then, these features are amalgamated in pairs for argumentative relation identification (ARI) subtask. Finally, the AC pairs and ascertained pertinent relations are employed for argumentative relation type classification (ARTC) subtask. However, these methods merely rely on a shared encoder to implicitly capture the interactions of the three subtasks, which cannot explicitly and comprehensively model the inter-relationship among subtasks. In this paper, we propose a novel method PITA for PromptIng Task interAction to model the inter-relationships among the three subtasks within a generative framework. Specifically, we employ a dynamic prompt template to indicate all ACs and AC pairs in the three subtasks. Then, we construct an undirected heterogeneous graph to capture the various relationships within and between ACs and AC pairs. We apply the Relational Graph Convolutional Network (RGCN) on the graph and inject the task interaction information into the soft prompts with continuous representations. PITA jointly decodes all ACs and AC pairs using the prompt template with task interaction information, which thus explicitly and comprehensively harmonizes the information propagation across the three subtasks. Extensive experiments show PITA achieves state-of-the-art performances on two AM benchmarks.

ID	ACs	Type	Pairs	Relation Type	ST Confidence
1	However, despite the fact that advertisements can be falseful and exaggerated.	Premise	(1,2)	Attack	√ 0.999
			(1,3)	-	-
2	It is also true that it plays a important role economically.	Claim	(1,4)	-	-
			(1,5)	-	-
3	They introduce new products , and provide various choices , not to mention they are often visually enjoyable.	Premise	(2,3)	Support	√ 0.941
			(2,4)	Support	✕ 0.005
4	Advertising also helps to keep prices at reasonable levels, creates a fair environment for commercial competition.	Premise	(2,5)	-	-
			(3,4)	-	-
5	Otherwise the market may be monopolistic , and the price as well as the choice will not be so selective.	Premise	(3,5)	Support	✕ 0.061
			(4,5)	-	-

Figure 1: An exemplary argumentative text from the PE dataset, where *Premise* and *Claim* denote the types of ACs. *Attack* and *Support* refer to the types of ARs. The ST confidence denotes the results of the ARI subtask.

1 Introduction

Argumentation mining (AM) (Lawrence and Reed, 2020) aims to detect argumentation structures in an argumentation text by identifying the arguments and the relations between them. Generally, AM involves four subtasks, including (1) Argument component segmentation (ACS) which extracts argument components (ACs) from an argumentative text; (2) argument component type classification (ACTC) that classifies the type of each AC (i.e., *Claim* or *Premise*); (3) argumentative relation identification (ARI) that identifies the argumentative relation (i.e., *Relevant* and *No-Relevant*) of AC pairs; (4) argumentative relation type classification (ARTC) that determines the type of the ARs (i.e., *Support* and *Attack*). We follow previous works (Potash et al., 2017; Kuribayashi et al., 2019; Bao et al., 2021a; Morio et al., 2022) and assume that the first subtask ACS has been completed, that is, ACs have been segmented, and focus on the other three subtasks. Figure 1 shows an example of an argumentative text and its structure, where the text is decomposed into five ACs and contains four AC pairs with ARs.

[*] Equal Contribution.

[†] Bin Liang, Min Yang and Ruifeng Xu are corresponding authors.

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 5036–5049
August 11-16, 2024 ©2024 Association for Computational Linguistics

Previous works (Kuribayashi et al., 2019; Bao et al., 2021a; Morio et al., 2022) usually employed multi-task learning to capture the relationships among the three subtasks, which have achieved remarkable progress for AM. Most of them utilized sequential decoding. Specifically, previous methods first separately tackled the ACs for ACTC. Then, they exploited the AC representations to model the relation between AC pairs for ARI. Finally, the AC pairs with identified relevant relations were used to classify the relation types for ARTC.

However, these methods, merely relying on a shared encoder, only implicitly capture task interactions and limit explicit, comprehensive inter-task modeling. Concretely, in sequential decoding, the information only explicitly flows from the ACTC decoder to ARI and ARTC decoders, not vice versa, potentially misguiding subsequent predictions if the former task decoding is inaccurate. For instance, the state-of-the-art model ST (Morio et al., 2022) misclassified the AC2 and AC4 as two Claims with a high degree of confidence. The pair decoder for ARI identifies the relation of the AC pair (AC2, AC4) as Non-Relevant, which is an incorrect answer, misleading the relation type for the ARTC subtask. The imbalanced information flow (i.e., from ACs in ACTC to AC pairs in ARI and ARTC as shown in Figure 2(a)) might have induced this error. Therefore, we argue that it may be important to model the inter-relationships among ACTC, ARI and ARTC for AM.

To address the aforementioned issue, we propose a novel method PITA for PromptIng Task interActions to model the inter-relationships among ACTC, ARI and ARTC. Our framework employs a generative encoder-decoder pre-trained language model (PLM), where the encoder focuses on the representation learning of ACs and AC pairs in the input text. The decoder equipped with a joint feature decoding mechanism facilitates the learning of task interaction patterns.

We recognize that subtasks are interconnected by the interactions among these task-specific representations of ACs or AC pairs. Specifically, to explicitly model the inter-relationship among subtasks and prompt the PLM, we first devise a dynamic prompt template to indicate all ACs and AC pairs. Considering these multiplex inter-relationships among ACs and AC pairs for the three subtasks, we construct an undirected heterogeneous graph. In the graph, the ACs and AC pairs serve as nodes, while their interrelations form the edges. Then, we apply Relational Graph Convolutional Networks (RGCN) (Schlichtkrull et al., 2018) over the graph to capture task interaction information among ACs and AC pairs. Finally, we inject the information into the soft prompt with continuous representation as the input of the decoder for jointly decoding all ACs and AC pairs for ACTC, ARI and ARTC simultaneously, rather than in separate steps.

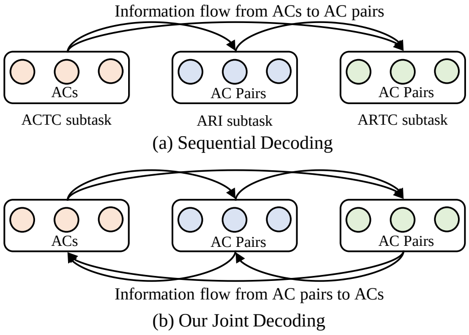


Figure 2: Comparison of information flow between sequential decoding and joint decoding.

The joint decoding mechanism explicitly incorporates task interactions, addressing the issues arising from sequential decoding. This method achieves equilibrium in the dissemination of information between individual ACs and their corresponding AC pairs. As shown in Figure 2(b), it not only utilizes inherent information existing amidst these ACs in ACTC for AC pairs in ARI and ARTC like previous methods (i.e., information flow from AC to AC pairs). But also it makes an AC assign heightened attention to evaluating its suitability for integration into an AC pair, transcending the scope of its self-contained informational content (i.e., information flow from AC pairs to ACs). In addition, different from the vanilla autoregressive generation paradigm (Lewis et al., 2019), our prompt template with the joint decoding mechanism only needs to be fed into the decoder one time and processed in parallel on GPUs.

In summary, our main contributions are as follows. (1) We introduce a novel method termed Task Interaction-Based Prompt Tuning (PITA) for AM. It employs prompt tuning in a generative framework catering to the multifaceted demands of multi-task AM. (2) We introduce a graph-based method to learn task interaction and information injection, devised to effectively represent the multifaceted relationships that exist between ACs and their respective pairs. (3) We conduct extensive experiments on two AM benchmarks. Experimental results exhibit significantly better performance for AM.

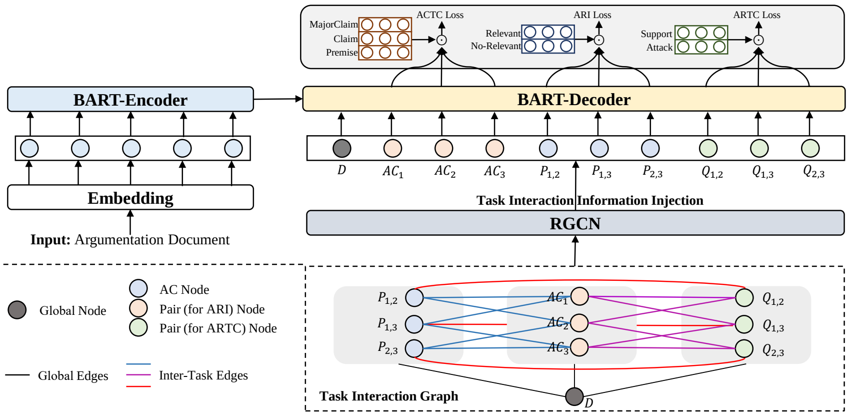


Figure 3: The architecture of PITA, where we omit the intra-task edges for concision.

2 Task Definition

Given the input argumentative text $W = \{w_1, w_2, \ldots, w_n\}$, the goal of AM is to identify a series of ACs $X = (x_1, x_2, \ldots, x_m)$ from it. Following previous work (Kuribayashi et al., 2019; Morio et al., 2022), we assume that the text span (b_i^1, b_i^2) of each AC x_i is given, which contains the beginning word index b_i^1 and the ending word index b_i^2. Specifically, for each AC x_i, an AM system first needs to predict its AC type $y_i^{\text{ACTC}} \in Y_{\text{ACTC}}$ (e.g., *Claim* and *Premise*). Then, it identifies the AR $y_{(i,j)}^{\text{ARI}} \in Y_{\text{ARI}}$ (e.g., *Rel* and *No-Rel*) between two ACs x_i and x_j. Finally, the type of each AR $y_{(i,j)}^{\text{ARTC}} \in Y_{\text{ARTC}}$ (e.g., *Support* and *Attack*) needs to be classified.

3 Methodology

Our proposed method, PITA, employs BART (Lewis et al., 2019) as the base model, and augments it with task interaction. BART is a standard Transformer-based PLM consisting of an encoder and a decoder: $\text{BART} = [\text{BART}_{\text{enc}}, \text{BART}_{\text{dec}}]$. The encoder is employed for representation learning of ACs and AC pairs, and the decoder is used for task interaction pattern learning among ACTC, ARI and ARTC. To enable BART to learn task interaction patterns, we first design a task interaction prompt template for the input text, as described in Section 3.2.1. The prompt template contains placeholders, which are essentially learnable vectors, serving as the reserved interface for task interaction information. Then, a task interaction graph is constructed and modeled through an RGCN (Section 3.2.2). The resulting node representations are integrated into BART_{dec} by the placeholders of the prompt template. In this way, the decoding process can be guided by the task interaction pattern (Section 3.2.3). Figure 3 illustrates the architecture of PITA.

3.1 Text Encoding

Given a piece of argumentative text W with AC spans, this module generates the AC and AC pair representations. Specifically, we feed W into BART-Encoder to obtain the context representation matrix $H^W \in \mathbb{R}^{n \times d}$, where d denotes the dimension of hidden states of BART.

$$H^W = \text{BART}_{\text{enc}}(W) \tag{1}$$

We use mean-pooling operation over H^W to obtain the input text representation $\mathbf{h}^W$. The representation of each AC x_i is derived by mean-pooling over H^W: $\mathbf{h}_i = \frac{1}{b_i^2 - b_i^1 + 1} \sum_{k=b_i^1}^{b_i^2} H_k^W$. Then, the representation of each AC pair is calculated by averaging the representations of the two ACs in the pair: $\mathbf{h}_{(i,j)} = (\mathbf{h}_i + \mathbf{h}_j)/2$. Subsequently, the output of the BART-encoder will be used as the input of the BART-decoder.

3.2 Task Interaction Prompt

To facilitate task interaction learning, we inject continuous prompt tokens into the decoder of BART.

3.2.1 Prompt Template

We devise a task interaction prompt template to produce the continuous prompt tokens for BART. In the prompt template, we employ different placeholders [1] for different ACs in the input text. Although both the ARI and ARTC subtasks use AC

[1] The placeholder, also known as virtual word, is implemented using specific tokens like the eos token $< s >$ in the BART vocabulary, and possess learnable vectors.

pairs to complete objectives, the task goals are different (i.e., relation identification for ARI and relation type classification for ARTC). We use two separate placeholders for the same AC pair for ARI and ARTC. In particular, given an argumentative text W with m ACs for ACTC, $m \times (m-1)/2$ AC pairs for ARI and $m \times (m-1)/2$ AC pairs for ARTC, the prompt template can be defined as:

$$\begin{aligned} &T = Concat([D], T_{ACTC}, T_{ARI}, T_{ARTC}) \\ &T_{ACTC} = [AC_1] \dots [AC_m] \\ &T_{ARI} = [P_{(1,2)}] \dots [P_{(m-1,m)}] \\ &T_{ARTC} = [Q_{(1,2)}] \dots [Q_{(m-1,m)}] \end{aligned}$$

where $[D]$ denotes the input text placeholder, providing global information (e.g., topics) for subsequent ACs and AC pair placeholders. This also proves effective in other tasks such as dialogue generation (Li et al., 2021). $[AC_i]$, $[P_{(i,j)}]$ and $[Q_{(i,j)}]$ denote the placeholders of AC x_i, AC pair (x_i, x_j) and AC pair (x_i, x_j) for ACTC, ARI and ARTC subtasks, respectively. Instead of using a fixed number of template tokens for all instances, our dynamic prompt template has different template tokens for different instances. Each token is infused with knowledge specific to these subtasks and designed to facilitate interaction among subtasks.

3.2.2 Task Interaction Graph Construction

Since the BART-decoder is an autoregression module, it only constructs the directed interaction of left-to-right among the prompt tokens and lacks their mutual interactions. To effectively capture the task interactions and balance the information propagation between ACs and AC pairs for the three subtasks, we construct an undirected heterogeneous graph.

The heterogeneous graph has four kinds of nodes corresponding to the placeholders in the prompt template: the input text placeholder $[D]$ as the global node, the AC placeholders $[AC_*]$ as the AC nodes and the two AC pair placeholders $[P_*]$ and $[Q_*]$ as the two kinds of AC pair nodes. Moreover, there are three kinds of inter-node edges in our graph:

- Global Edge: The global node is connected to all other nodes. It can transmit the global information in the input text to other nodes.

- Intra-Task Edge: In each subtask, all nodes are fully connected, i.e., AC nodes connect AC nodes in ACTC and AC pair nodes connect AC pair nodes in ARI (ARTC). These edges can help the AC/pair nodes access contextual information.

- Inter-Task Edge: Two nodes from two different subtasks are connected if they have an inclusion relationship. For example, the AC pair node $[P_{(i,j)}]$ ($[Q_{(i,j)}]$) in ARI (ARTC) links individual AC nodes $[AC_i]$ and $[AC_j]$ as the AC pair (x_i, x_j) includes AC x_i and x_j. The AC pair node $[P_{(i,j)}]$ in ARI is connected with the AC pair node $[Q_{(i,j)}]$ in ARTC. They are the primary way for ACs and AC pairs to interact with each other. It not only helps the AC pair nodes to invoke the inherent information of ACs. But also the AC nodes can interact with AC pair nodes to transmit argumentation relation information of AC pairs to themselves.

3.2.3 Task Interaction Information Injection

To make full use of the graph-based task interaction pattern, we inject the task interaction information into template embedding [2]. Specifically, we first feed the whole prompt template into the embedding layer of BART, which embeds input sequence T into the embedding space $E^T = [\mathbf{e}^D, \mathbf{e}_1, \dots, \mathbf{e}_m, \mathbf{e}^P_{(1,2)}, \dots, \mathbf{e}^P_{(m-1,m)}, \mathbf{e}^Q_{(1,2)}, \dots, \mathbf{e}^Q_{(m-1,m)}]$, where $\mathbf{e}^D$ denotes the embedding of input text placeholder $[D]$. $\mathbf{e}_i$, $\mathbf{e}^P_{(i,j)}$ and $\mathbf{e}^Q_{(i,j)}$ are the embeddings of AC and AC pair placeholders $[AC_i]$, $[P_{(i,j)}]$ and $[Q_{(i,j)}]$s, respectively.

Then, we apply an RGCN (Schlichtkrull et al., 2018) on our undirected heterogeneous graph to model the task interaction pattern. Given a node $u \in T$ at the l-th RGCN layer, the information interaction and aggregation operation is defined as follows:

$$\mathbf{e}_u^{l+1} = ReLU(\mathbf{e}_u^l + \sum_{r \in R} \sum_{v \in N_r(u)} \frac{1}{|N_r(u)|} W_r^l \mathbf{e}_v^l + b_r^l) \quad (2)$$

where $N_r(u)$ denotes the neighbors for node u connected with the edge of type r, ReLU is the ReLU activation function, W_r^l and b_r^l are the trainable parameters. For the first RGCN layer, we adopt the embeddings of placeholders in the template T to initialize node features. Finally, we select the representation of all nodes of the last layer L as the

[2] We also consider directly putting the template embedding into BART-Decoder and take the output hidden states of the decoder as the input of the RGCN to incorporate task interaction information. The detailed analysis can be seen in the ablation study.

updated prompt template representations, which aggregate task interaction information from a certain heterogeneous graph.

Although the prompt and context information would interact with each other at the cross-attention layers in the decoder, simply feeding the updated template placeholder features into the decoder makes it hard to focus accurately on the correct segment of input text for ACs and AC pairs. Therefore, we add the context representations of ACs and AC pairs to the updated template representations to obtain the context-specific prompt embeddings $\hat{E}^T = [\hat{\mathbf{e}}^D, \hat{\mathbf{e}}_1, \ldots, \hat{\mathbf{e}}_m, \hat{\mathbf{e}}^P_{(1,2)}, \ldots, \hat{\mathbf{e}}^P_{(m-1,m)}, \hat{\mathbf{e}}^Q_{(1,2)}, \ldots, \hat{\mathbf{e}}^Q_{(m-1,m)}]$, where $\hat{\mathbf{e}}^D = \mathbf{e}^{D,L} + \mathbf{h}^W$, $\hat{\mathbf{e}}_i = \mathbf{e}^L_i + \mathbf{h}_i$, $\hat{\mathbf{e}}^k_{(i,j)} = \mathbf{e}^{k,L}_{(i,j)} + \mathbf{h}_{(i,j)}$ and $k \in \{P, Q\}$.

After that, we feed the prompt representations into the BART decoder to get the output hidden states $\mathbf{h}^0_i$, $\mathbf{h}^1_{(i,j)}$ and $\mathbf{h}^2_{(i,j)}$ of the placeholders $[AC_i]$, $[P_{(i,j)}]$ and $[Q_{(i,j)}]$. Recall the joint decoding process of ACs and AC pairs from the prompt. It is obvious that the interaction among different subtasks is considered under this paradigm. In this way, our method can solve the problems in sequential decoding and balance the information between ACs and AC pairs.

3.3 Objective Function

The training objective of PITA is to generate outputs that replace the placeholders in the prompt template with gold labels. For ACTC, PITA is expected to derive the argumentation type of the AC x_i using the hidden state of placeholder $[AC_i]$. PITA employs the placeholder $[P_{(i,j)}]$ to determine the relation between AC pair (x_i, x_j) for ARI. For ARTC, PITA uses the placeholder $[Q_{(i,j)}]$ to predict the relation type between AC pair (x_i, x_j). Formally, we create a learnable label word vector $\mathbf{v}_i$ for each label y_i in each subtask. We utilize cross-entropy functions as the learning objectives of the ACTC, ARI and ARTC subtasks, which are defined as follows:

$$\begin{aligned}
\mathcal{L}_{ACTC} &= -\sum_{i=0}^{m} log(\hat{y}_i^{\text{ACTC}}) \\
\mathcal{L}_{ARI} &= -\sum_{i=0}^{m}\sum_{j=i+1}^{m} log(\hat{y}_{(i,j)}^{\text{ARI}}) \\
\mathcal{L}_{ARTC} &= -\sum_{i=0}^{m}\sum_{j=i+1}^{m} log(\hat{y}_{(i,j)}^{\text{ARTC}})
\end{aligned} \tag{3}$$

where $\hat{y}_i^{\text{ACTC}} = \mathbf{v}^0_k \mathbf{h}^0_i$, $\hat{y}_{(i,j)}^{\text{ARI}} = \mathbf{v}^1_k \mathbf{h}^1_{(i,j)}$ and $\hat{y}_{(i,j)}^{\text{ARTC}} = \mathbf{v}^2_k \mathbf{h}^2_{(i,j)}$ are the predicted probability of ground truth labels y_k of the AC x_i for ACTC, AC pair (x_i, x_j) for ARI and AC pair (x_i, x_j) for ARTC subtasks, respectively. $\mathbf{v}^0_k$, $\mathbf{v}^1_k$ and $\mathbf{v}^2_k$ are the k-th label vector in ACTC, ARI and ARTC.

We train PITA by jointly optimizing the three subtasks. The total training object is defined as follows:

$$\mathcal{L} = \mathcal{L}_{\text{ACTC}} + \mathcal{L}_{\text{ARI}} + \mathcal{L}_{\text{ARTC}} \tag{4}$$

3.4 Efficiency Considerations

PITA has three components including a BART, a RGCN and a prompt template. The RGCN is required to model the task interaction graph, which has a time complexity $O(m^4)$ and could cause efficiency considerations. In practice, this issue is minor for our experiments on the two datasets (PE and CDCP) and real scenarios, because the number [3] of ACs in argumentation texts is usually relatively small scales. Noted our prompt template with the joint decoding mechanism only needs to be fed into the decoder one time and processed in parallel. It is more efficient compared with the autoregressive paradigm (Lewis et al., 2019) which generates tokens sequentially during inference. We present the time cost of PITA in Appendix A.4.

4 Experimental Setup

Datasets To evaluate the effectiveness of our PITA model, we conduct extensive experiments on two widely-used AM datasets, **PE** (Stab and Gurevych, 2017) and **CDCP** (Park and Cardie, 2018) following the official split. The detailed statistics of PE and CDCP are summarized in Appendix A.1.

Evaluation Metrics We employ the same evaluation metrics with the previous works (Kuribayashi et al., 2019; Liu et al., 2022), including F_1 score and macro averaged score (denoted as Macro). We adopt the macro averaged score for ACTC and ARTC and calculate F_1 scores for determining the relevant (Rel) AR between ACs for ARI following (Morio et al., 2022).

Baselines We compare PITA with the following strong baseline models. Following previous works (Morio et al., 2022), for the PE dataset, we compare our model with six strong baselines, including **Joint-ILP** (Stab and Gurevych, 2017),

[3] The average number of ACs in the PE and CDCP samples is 3.5 and 6.5, respectively.

Model	ACTC	ARI	ARTC	Avg
Joint-ILP	82.6	58.5	-	-
Joint-PN	84.9	60.8	-	-
BERT-Trans	88.4	70.6	-	-
LSTM+dist	85.7	67.8	54.3	69.3
BART	82.7	62.9	53.4	66.3
ST	86.8	69.3	57.1	71.1
PITA (our)	**88.3**	**73.5**	**59.2**	**73.7 (+2.6)**

Table 1: Perfromance comparison on the PE dataset in terms of Macro for ACTC and ARTC, as well as F_1 for ARI. **Avg** indicates the average value across all metrics. Our improvements over baselines are statistically significant with p $<$ 0.05.

Model	ACTC	ARI	ARTC	Avg
SSVM-strict	73.2	26.7	-	-
TSP-PLBA	78.9	34.0	-	-
BERT-Trans	82.5	37.3	-	-
DR-LG	65.3	29.3	15.0	36.5
BART	81.4	32.9	16.7	43.7
ST	82.3	40.2	20.4	47.6
PITA (our)	**83.6**	**44.9**	**23.8**	**50.8 (+3.2)**

Table 2: Performance comparison on CDCP dataset in terms of Macro for ACTC and ARTC, as well as F_1 for ARI. **Avg** indicates the average value across all metrics. Our improvements over baselines are statistically significant with p $<$ 0.05.

Joint-PN (Potash et al., 2017), **LSTM+dist** (Kuribayashi et al., 2019), **BERT-Trans** (Bao et al., 2021a) **BART** (Lewis et al., 2019) and **ST** (Morio et al., 2022). For the CDCP dataset, we compare our model with five strong baselines, which are **DR-LG** (Galassi et al., 2018), **SSVM-strict** (Niculae et al., 2017), **TSP-PLBA** (Morio et al., 2020), **BERT-Trans** (Bao et al., 2021a), **BART** (Lewis et al., 2019) and **ST** (Morio et al., 2022).

Implementation Details We use PyTorch to implement the proposed framework based on Transformers (Wolf et al., 2020) on NVIDIA TESLA A100-PCIE-40GB. Our model is optimized using AdaW (Loshchilov and Hutter, 2017) with the learning rates of 3e-5 and weight decay of 1e-2 on both PE and CDCP datasets. We set the batch size to 4 on both PE and CDCP datasets. For both datasets, we adopt dropout (Srivastava et al., 2014) with a dropout rate of 0.1 to avoid overfitting. We set the layer number L of RGCN to 1 because of its best performance. All experiments are performed 5 times with different random seeds, and the evaluation scores are averaged. Our code is available at `https://github.com/syiswell/PITA`.

Model	PE			CDCP		
	ACTC	ARI	ARTC	ACTC	ARI	ARTC
PITA	**88.3**	**73.5**	**59.2**	83.6	**44.9**	**23.8**
w/o TIG	85.5	70.6	56.6	82.5	40.8	21.5
PB TIG	86.4	69.5	55.8	83.6	41.1	21.5
w/o IRT-Edge	87.3	71.5	57.9	83.2	41.7	21.8
w/o IET-Edge	86.8	71.6	57.1	82.8	41.4	22.6
w/o G-Edge	86.7	71.1	58.4	**83.7**	41.2	22.4

Table 3: The impact of different components in terms of Macro for ACTC and ARTC, as well as F_1 for ARI.

5 Experimental Results

5.1 Performance Comparison

The comparison results on PE and CDCP datasets are summarized in Table 1 and Table 2, respectively. We can observe that PITA achieves the best performance on both datasets. On the PE dataset, our PITA model outperforms state-of-the-art (SOTA) model ST by 4.2% and 2.1% on ARI and ACTC subtasks in terms of F_1 and Macro, respectively. We observe similar trends for the CDCP dataset. Our model brings 4.7% improvement of F_1 score for ARI and 3.4% improvement of Macro score for ARTC. We argue that task interaction learning by joint decoding manner plays an important role in PITA, balancing the information obtained by ACs and AC pairs.

In addition, we also observe that the feature-based models (i.e., Joint-ILP and St-SVM-strict) perform poorly since they heavily rely on feature engineering. The BERT- and Longformer-based neural network (i.e., BERT-Trans and ST) consistently outperforms the LSTM-based baselines (i.e., LSTM+dist and TSP-PLBA). This may be because BERT and Longformer-based methods can exploit rich knowledge from PLMs trained on large-scale general corpora. Our PITA model performs better than the strong BERT- and Longformer-based models by modeling task interaction patterns to effectively capture the rich dependency information within and across the ACs and AC pairs.

5.2 Ablation Study

To analyze the impact of different components in PITA, we conduct ablation studies and report the results in Table 3. We can observe that by removing the whole task-interaction graph (w/o TIG), the model cannot access the connectivity of the ACs and AC pairs, with significant performance degradation. It is worth noting that removing the whole task-interaction graph (w/o TIG) is equal to only using the prompt template, which is ultimately close

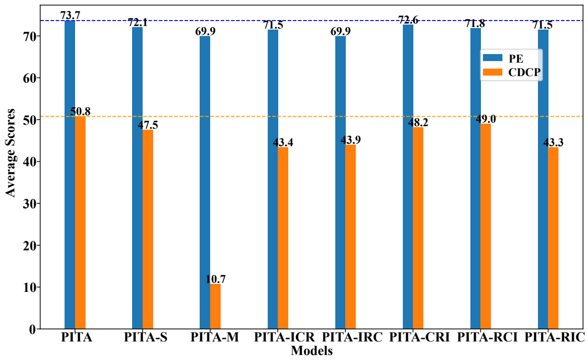


Figure 4: The impact of different templates in terms of average score over ACTC, ARTC and ARI. Note the template of PATA-M exceeds the maximum length of BART on CDCP.

Data	Type	Model	ACTC	ARI	ARTC
PE	FT	PITA	88.3	73.5	59.2
		LLAMA2-FT	83.0	59.0	50.8
		LLAMA2-PITA	87.2	70.4	53.6
		-w/o TIG	85.9	69.1	52.8
	FS	ChatGPT-3.5-turbo	64.2	57.4	38.6
		LLAMA2	23.2	10.8	7.6
CDCP	FT	PITA	83.6	44.9	23.8
		LLAMA2-FT	77.6	27.1	14.4
		LLAMA2-PITA	85.5	35.1	17.8
		-w/o TIG	84.0	31.5	16.0
	FS	ChatGPT-3.5-turbo	57.4	22.6	11.3
		LLAMA2	23.7	10.3	5.3

Table 4: Performance comparison on PE and CDCP dataset in terms of Macro for ACTC and ARTC, as well as F_1 for ARI. FT and FS represent fine-tuning and few-shot learning approaches, respectively.

to vanilla BART. In addition, to further illustrate the effectiveness of task interaction information injection, we put the task interaction graph behind the BART-Decoder and the hidden states of the decoder as the input of the RGCN (denoted by PB TIG). We can observe that PB TIG leads to a significant decrease in performance. This is because the PB TIG separately decodes the representations of ACs and AC pairs in the decoder, which cannot efficiently utilize the knowledge within the PLM to obtain high-quality AC and AC pair representations.

To examine the importance of different components in the task interaction graph, we consider removing the different nodes and edges. We can see that after removing the global node with the global edge (w/o G-Edge), the model performance drops, showing the effectiveness of the global information for ACs and AC pairs. Removing the intra-task edge (w/o IRT-Edge) degrades the performance, verifying that contextual information is important for AM. Removing inter-task edge (w/o IET-Edge) leads to performance drops, demonstrating that task interaction pattern learning facilitates a balanced propagation of information between ACs and their corresponding AC pairs.

5.3 Adaptability Experiment

Inspired by the recent success of Large Language Models (LLMs) (Min et al., 2023), we conduct additional experiments on two prominent LLMs, namely ChatGPT-3.5-Turbo and LLAMA2. We employ the natural language prompt approach like (Madaan et al., 2022; Li et al., 2023) and few-shot in-context learning (here 3-shot is used due to the limitation of input length) in ChatGPT-3.5-Turbo and LLAMA2 for AM. Figure 6 displays the format of the natural language prompt for AM. To evaluate the adaptability of our method, we adapt our approach to fine-tuning LLAMA2 using LORA (Hu et al., 2021) (denoted by LLAMA2-PIAT). Besides, we fine-tune LLAMA2 with the natural language prompt approach (denoted by LLAMA2-FT) and LLAMA2-PIAT without the task interaction graph using LORA as baselines. Each fine-tuning method uses a batch size of 1. The results are presented in Table 4.

We observe that few-shot-based methods (i.e., LLAMA2 and ChatGPT-3.5-Turbo) significantly underperform fine-tuning methods (i.e., PITA, LLAMA2-PITA and LLAMA2-FT) on PE and CDCP. Among these fine-tuning approaches, our PITA framework (LLAMA-PITA) outperforms the w/o TIG as well as the LLAMA-FT which is fine-tuned using natural language prompt, validating the effectiveness of our task interaction learning. In addition, LLAMA-PITA performs worse than PITA using BART as the base model. This is because LLAMA-PITA has a large number of parameters (including trainable and non-trainable parameters) while the AM datasets have a small number of samples, which leads to severe overfitting.

5.4 Impact of Different Templates

We explore how different types of prompts affect the performance in this section, as shown in Figure 4. We first compare two template variants with our method: PITA-S, which uses an identical placeholder for different ACs (AC pairs) in each subtask, and PITA-M, where each AC and AC pair followed an *[MASK]* token for prediction, simulating the pretrained objective of BART. Furthermore, We argue that there are task order biases in our PITA model due to the autoregressive generative paradigm. Thus, we investigate the effect of task order in the template and explore five variants (denoted by PITA-ICR, PITA-IRC, PITA-CRI, PITA-RCI and PITA-RIC) that rearrange the placeholder positions for the ACTC (C), ARI (I), and ARTC (R) tasks. The formats of all prompt templates are detailed in Table 7.

We find that (1) our template outperforms other template variants; (2) PITA-S performs worse than ours. With the same placeholders in each subtask, where the template neither contains specific knowledge about ACs or AC pairs nor learns task interaction patterns during training; (3) PITA-M achieves worse results than ours, demonstrating our ACs and AC pairs placeholders can be better used for task-specific prediction by injecting task interaction information, whereas the extra *[MASK]* simply contains the priori knowledge learned during pretraining; (4) Unsurprisingly, PITA-ICR, PITA-IRC, PITA-CRI, PITA-RCI and PITA-RIC obtain worse performance than ours, which verifies that the task order of ACTC, ARI and ARTC is better setting. This setting is widely used in previous works.

5.5 Impact of Different Connections of Task Interaction Graph

In the task interaction graph, we connect nodes according to the strategy described in Section 3.2.2, but this connection is not unique. To assess the effectiveness of the proposed connection strategy, we experiment with two different variants. The first variant, termed Random connection (RC), randomly links two nodes. The second variant, named Full connection (FC), links each node pair. The results are presented in Table 5. RC outperforms FC yet falls short of our strategy, indicating that indiscriminate node connections can introduce noisy information. Our connection strategy considers not only the contextual information within each subtask and the interactions across the three subtasks. This ensures a balanced information flow between Argument Components (ACs) and AC pairs, leading to superior results.

5.6 Case Study

We analyze two examples selected from the benchmark corpus to demonstrate the effectiveness of task interaction for balancing information among three subtasks, which is shown in Figure 5. Our task interaction graph makes ACs and AC pairs interact with each other. It not only helps the AC pair nodes to invoke the type information of ACs and itself. But also the AC nodes can interact with AC pair nodes to transmit argumentation relation information of AC pairs to themselves. In the first example, although ST classifies all the correct ACs, it cannot couple the correct pair (AC3, AC4). By considering the argumentation relationship among ACs, PITA avoids this situation. For the second example, ST and PITA both classify the wrong type for AC4 in ACTC. In ARTC, the sequential encoding method ST cannot couple the pair (AC2, AC4), resulting in one undetected pair. On the contrary, PITA can avoid them and correct the pair type by task interaction pattern learning and the balance of information flow between AC and AC pairs.

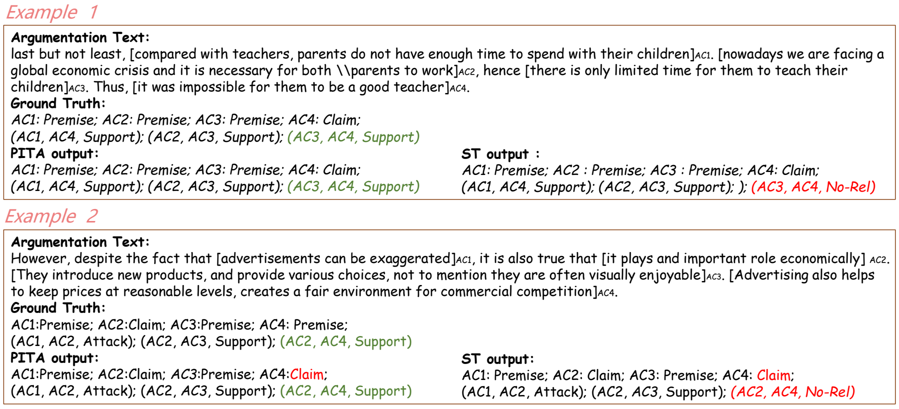
Model	PE			CDCP		
	ACTC	ARI	ARTC	ACTC	ARI	ARTC
PITA	**88.3**	**73.5**	**59.2**	**83.6**	**44.9**	**23.8**
with RC	86.9	72.3	56.7	82.2	43.5	21.0
with FC	86.8	71.6	57.4	79.2	42.2	21.4

Table 5: The impact of different connections of task interaction graph.

6 Related Works

6.1 Argumentation Mining

Argumentation mining (AM) aims to identify and extract the argumentation structures from argumentative texts automatically (Lawrence and Reed, 2020; Bao et al., 2021b; Cheng et al., 2021; Sun et al., 2022; Guo et al., 2023; Chen et al., 2023). Early works (Peldszus and Stede, 2015; Persing and Ng, 2016; Stab and Gurevych, 2017; Afantenos et al., 2018) applied methods like minimum spanning trees (MST) and integer linear programming (ILP) with discrete features, focusing heavily on feature engineering, which is both labor-intensive and time-consuming. With the rise of deep learning, Potash et al. (2017) introduced a sequence-to-sequence model with pointer networks and attention for AM. Kuribayashi et al. (2019) added linguistic clues to enhance AM performance, while Niculae et al. (2017) used SVM-based structured learning for both tree and non-tree data, albeit requiring argumentation-specific factor graph designs. Galassi et al. (2018) and Morio et al. (2020) introduced residual networks and task-specific modules for AM, tackling tree and non-tree structures separately. Bao et al. (2021a) proposed a neural transition-based model using BERT for



Example 1

Argumentation Text:
last but not least, [compared with teachers, parents do not have enough time to spend with their children]$_{AC1}$. [nowadays we are facing a global economic crisis and it is necessary for both \\parents to work]$_{AC2}$, hence [there is only limited time for them to teach their children]$_{AC3}$. Thus, [it was impossible for them to be a good teacher]$_{AC4}$.

Ground Truth:
AC1: Premise; AC2: Premise; AC3: Premise; AC4: Claim;
(AC1, AC4, Support); (AC2, AC3, Support); (AC3, AC4, Support)

PITA output:
AC1: Premise; AC2: Premise; AC3: Premise; AC4: Claim;
(AC1, AC4, Support); (AC2, AC3, Support); (AC3, AC4, Support)

ST output :
AC1: Premise; AC2 : Premise; AC3 : Premise; AC4: Claim;
(AC1, AC4, Support); (AC2, AC3, Support);); (AC3, AC4, No-Rel)

Example 2

Argumentation Text:
However, despite the fact that [advertisements can be exaggerated]$_{AC1}$, it is also true that [it plays and important role economically] $_{AC2}$. [They introduce new products, and provide various choices, not to mention they are often visually enjoyable]$_{AC3}$. [Advertising also helps to keep prices at reasonable levels, creates a fair environment for commercial competition]$_{AC4}$.

Ground Truth:
AC1:Premise; AC2:Claim; AC3:Premise; AC4: Premise;
(AC1, AC2, Attack); (AC2, AC3, Support); (AC2, AC4, Support)

PITA output:
AC1:Premise; AC2:Claim; AC3:Premise; AC4:Claim;
(AC1, AC2, Attack); (AC2, AC3, Support); (AC2, AC4, Support)

ST output:
AC1: Premise; AC2: Claim; AC3: Premise; AC4: Claim;
(AC1, AC2, Attack); (AC2, AC3, Support); *(AC2, AC4, No-Rel)*

Figure 5: Examples predicted by PITA and ST, where incorrect prediction results are marked in red.

ACTC and transforming ARI into action prediction, and Morio et al. (2022) used a Longformer-based model with biaffine function for AM. Different from previous works, we explicitly model the task interactions by constructing the complex relations within and between ACs and AC pairs for AM through a generative framework.

6.2 Prompt Tuning

Prompt tuning (Liu et al., 2023) has attracted much attention in the field of natural language processing (NLP), such as text classification (Schick and Schütze, 2021; Wang et al., 2022), text generation (Li and Liang, 2021) and information extraction (Ma et al., 2022). Existing studies on prompt tuning learning mainly focus on discrete and continuous prompts. The former designs text-based prompts (Jiang et al., 2020; Gao et al., 2020; Schick and Schütze, 2020), while the latter prepend a learnable prompt vector to word embeddings (Lester et al., 2021). We adopt the latter because of its flexibility and extensiveness. To the best of our knowledge, our PITA is the first work to use prompt tuning within a generative framework for AM.

7 Conclusion

In this paper, we proposed a novel model PITA, which prompted task interaction to model the relationships within and between ACs and AC pairs through a joint decoding mechanism encompassed within a generative framework. To explicitly model the interaction among three subtasks, we devised a dynamic prompt template to prompt all ACs and AC pairs in the three subtasks. Then, we constructed an undirected heterogeneous graph to capture the comprehensive relationships within and between ACs and AC pairs. Experimental results on two benchmarks showed that our method outperformed strong baselines significantly.

Limitation

To point out future research direction for AM, we perform an error analysis on 100 cases where our PITA made mistakes for ACTC, ARI and ARC subtasks. In the ACTC subtask, we discover that there is a type bias for different positions of ACs. For instance, PITA tends to predict the first AC as “Claim” and the last AC as “Promise”. This is because PITA overfits the type distribution of ACs’ different positions based on the prompt placeholders. Maybe we can adopt a debias approach to alleviate this issue. For ARI, we find that PITA captures the connection between long-distance ACs so excessively that it identifies additional AC pairs. In the argumentation structure, AC pairs with relationships should not be too far apart, or it won’t follow the argumentation structure habits of human texts. Therefore, we suggest that during ARI tasks, incorporating distance loss between ACs can guide the model to focus more on the connection between ACs with moderate distances, which could potentially address the issue. In the ARTC task, the most serious problem is the error identification of AC pairs, resulting in incorrect AC pair type classification. In addition, there is a serious class imbalance problem that induces incorrect model predictions.

Last, we argue that there are order biases in our PITA model as described in Table 4 due to the autoregressive generative paradigm. In particular, the order of the placeholders in the prompt template is fixed, but there are actually no order relations between these placeholders. Although we mitigate the order bias between tasks using task interaction learning and experimentally validate the effectiveness of our task interaction learning. However, two directions worth exploring are combining multiple templates with different task orders to eliminate order bias between tasks and the fact that order bias within each task needs to be taken into account, even if it may be increases the complexity of the model. Therefore, the tradeoff between order modeling and complexity is what we need to explore. Additionally, a new method that is free from the influence of order bias is more desirable.

Acknowledgements

We thank the anonymous reviewers for their valuable suggestions to improve the quality of this work. This work was partially supported by the National Key Research and Development Program of China (2022YFF0902100), the National Natural Science Foundation of China (62176076), Natural Science Foundation of Guangdong (2023A1515012922), the Shenzhen Foundational Research Funding (JCYJ20210324115614039, JCYJ20220818102415032), the Major Key Project of PCL2021A06, Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies (2022B1212010005).

References

Stergos Afantenos, Andreas Peldszus, and Manfred Stede. 2018. Comparing decoding mechanisms for parsing argumentative structures. *Argument & Computation*, 9(3):177–192.

Jianzhu Bao, Chuang Fan, Jipeng Wu, Yixue Dang, Jiachen Du, and Ruifeng Xu. 2021a. 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.

Jianzhu Bao, Bin Liang, Jingyi Sun, Yice Zhang, Min Yang, and Ruifeng Xu. 2021b. Argument pair extraction with mutual guidance and inter-sentence relation graph. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3923–3934.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Guizhen Chen, Liying Cheng, Luu Anh Tuan, and Lidong Bing. 2023. Exploring the potential of large language models in computational argumentation. *arXiv preprint arXiv:2311.09022*.

Liying Cheng, Tianyu Wu, Lidong Bing, and Luo Si. 2021. Argument pair extraction via attention-guided multi-layer multi-cross encoding. 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 6341–6353.

Andrea Galassi, Marco Lippi, and Paolo Torroni. 2018. Argumentative link prediction using residual networks and multi-objective learning. In *Proceedings of the 5th Workshop on Argument Mining*, pages 1–10.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2020. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*.

Jia Guo, Liying Cheng, Wenxuan Zhang, Stanley Kok, Xin Li, and Lidong Bing. 2023. Aqe: Argument quadruplet extraction via a quad-tagging augmented generative approach. *arXiv preprint arXiv:2305.19902*.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Tatsuki Kuribayashi, Hiroki Ouchi, Naoya Inoue, Paul Reisert, 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.

John Lawrence and Chris Reed. 2020. Argument mining: A survey. *Computational Linguistics*, 45(4):765–818.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension.

Peng Li, Tianxiang Sun, Qiong Tang, Hang Yan, Yuanbin Wu, Xuanjing Huang, and Xipeng Qiu. 2023. CodeIE: Large code generation models are better few-shot information extractors. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Toronto, Canada. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Zekang Li, Jinchao Zhang, Zhengcong Fei, Yang Feng, and Jie Zhou. 2021. Conversations are not flat: Modeling the dynamic information flow across dialogue utterances. 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 128–138.

Junlong Liu, Xichen Shang, and Qianli Ma. 2022. Pair-based joint encoding with relational graph convolutional networks for emotion-cause pair extraction. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5339–5351.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Yubo Ma, Zehao Wang, Yixin Cao, Mukai Li, Meiqi Chen, Kun Wang, and Jing Shao. 2022. Prompt for extraction? paie: Prompting argument interaction for event argument extraction. *arXiv preprint arXiv:2202.12109*.

Aman Madaan, Shuyan Zhou, Uri Alon, Yiming Yang, and Graham Neubig. 2022. Language models of code are few-shot commonsense learners. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1384–1403.

Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heintz, and Dan Roth. 2023. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Computing Surveys*, 56(2):1–40.

Gaku Morio, Hiroaki Ozaki, Terufumi Morishita, Yuta Koreeda, and Kohsuke Yanai. 2020. Towards better non-tree argument mining: Proposition-level biaffine parsing with task-specific parameterization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3259–3266.

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. *arXiv preprint arXiv:1704.06869*.

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)*.

Andreas Peldszus and Manfred Stede. 2015. Joint prediction in mst-style discourse parsing for argumentation mining. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 938–948.

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.

Peter Potash, Alexey Romanov, and Anna Rumshisky. 2017. Here's my point: Joint pointer architecture for argument mining. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1364–1373.

Timo Schick and Hinrich Schütze. 2020. It's not just size that matters: Small language models are also few-shot learners. *arXiv preprint arXiv:2009.07118*.

Timo Schick and Hinrich Schütze. 2021. Exploiting cloze questions for few shot text classification and natural language inference.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*, pages 593–607. Springer.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Christian Stab and Iryna Gurevych. 2017. Parsing argumentation structures in persuasive essays. *Computational Linguistics*, 43(3):619–659.

Yang Sun, Bin Liang, Jianzhu Bao, Min Yang, and Ruifeng Xu. 2022. Probing structural knowledge from pre-trained language model for argumentation relation classification. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3605–3615.

Zihan Wang, Peiyi Wang, Tianyu Liu, Binghuai Lin, Yunbo Cao, Zhifang Sui, and Houfeng Wang. 2022. Hpt: Hierarchy-aware prompt tuning for hierarchical text classification. *arXiv preprint arXiv:2204.13413*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

A Appendices

Type	PE	CDCP
Paragraphs	1833	731
Train	1464	581
Test	369	150
Components	6089	4779
Relations	3832	1353
Components Per Sample	3.5	6.5

Table 6: The statistics of the PE and CDCP datasets.

A.1 Data statistics

We conduct experiments on two popular benchmark datasets.

- **PE** (Stab and Gurevych, 2017): The PE dataset contains 420 essays with 1833 paragraphs. There are three types of ACs (i.e., *MajorClaim* (MC), *Claim* and *Premise*) and two types of ARs (i.e., *Support* and *Attack*). Each AC has at most one outgoing AR so the argumentation graph of the paragraph can be either directed trees or forests. In addition, we extend each AC by including its argumentative marker following previous works (Kuribayashi et al., 2019; Bao et al., 2021a). We split this dataset into a training set of 1464 ACs and a testing set of 369 ACs, and randomly choose 10% of the training set as the validation set, which is consistent with previous works (Kuribayashi et al., 2019; Bao et al., 2021a).
- **CDCP** (Park and Cardie, 2018): The CDCP dataset contains 731 paragraphs. All ACs are classified into five types: *Value*, *Policy*, *Testimony*, *Fact* and *Reference*. The ARs have two types: *Reason* and *Evidence*. Each AC may have several outgoing ARs, thus the argumentation graph is of non-tree structure. The whole dataset is partitioned into a training set of 581 ACs and a testing set of 150 ACs. We randomly choose 10% of the training set as the validation set following (Bao et al., 2021a; Morio et al., 2022).

The statistics of the two datasets are summarized in Table 6.

A.2 Baselines

Following previous works (Morio et al., 2022), for the PE dataset with tree structure, we compare our model with six strong baselines:

- **Joint-ILP** (Stab and Gurevych, 2017): This model optimizes argument component types (ACTC) and argumentative relations (ARI) using Integer Linear Programming.
- **Joint-PN** (Potash et al., 2017): It applies a Pointer Network with attention mechanism to jointly learn ACTC and ARI.
- **LSTM+dist** (Kuribayashi et al., 2019): This work first introduce LSTM-minus-based span representation with pretrained ELMO embedding for of AM.
- **BERT-Trans** (Bao et al., 2021a): This model employs neural transition-based model by generating a sequence of actions for argumentation mining.
- **BART** (Lewis et al., 2019): The model only uses a BART and converts the AM into a sequence generative formulation. The format of the target sequence is similar to our prompt format.
- **ST** (Morio et al., 2022): This method employs Longformer (Beltagy et al., 2020) and biaffine function for AM.

For the CDCP dataset with non-tree structure, we compare our model with six strong baselines, which are:

- **DR-LG** (Galassi et al., 2018): This method explores the use of residual networks with link-guided training to jointly learn ACTC, ARI and ARTC.
- **SSVM-strict** (Niculae et al., 2017): This method is a variant of structured SVM with strict factor graph for both ACTC and ARI.
- **TSP-PLBA** (Morio et al., 2020): The model incorporates task-specific parameterization to encode ACs and proposition-level biaffine attention to capture the structure of argumentation corpus.
- **BERT-Trans** (Bao et al., 2021a): This model employs neural transition-based model by generating a sequence of actions for argumentation mining, which is also the current state-of-the-art method on the CDCP dataset.

Input:
The text is "Despite the fact that advertisements can be exaggerated, it is also true that it plays an important role economically. They introduce new products.".
The argumentation components in the text:
"advertisements can be exaggerated".
"it plays and important role economically".
"They introduce new products".
The type of argumentation component and argumentation relation in the text:

Output:
"advertisements can be exaggerated." is Premise.
"it plays an important role economically." is Claim.
"They introduce new products." is Premise.
the argument relation between "advertisements can be exaggerated." and "it plays an important role economically." is "Attack".
the argument relation between "it plays an important role economically." and "They introduce new products." is "Support".

Figure 6: The format of natural language prompts for AM.

Type	Templates
PITA	$[D][AC_1]\dots[AC_m][P_{(1,2)}]\dots[P_{(m-1,m)}][Q_{(1,2)}]\dots[Q_{(m-1,m)}]$
PITA-S	$[D][AC]\dots[AC][P]\dots[P][Q]\dots[Q]$
PITA-M	$[D][AC_1][MASK]\dots[AC_m][MASK]$ $[P_{(1,2)}][MASK]\dots[P_{(m-1,m)}][MASK]$ $[Q_{(1,2)}][MASK]\dots[Q_{(m-1,m)}][MASK]$
PITA-ICR	$[D][P_{(1,2)}]\dots[P_{(m-1,m)}][AC_1]\dots[AC_m][Q_{(1,2)}]\dots[Q_{(m-1,m)}]$
PITA-IRC	$[D][P_{(1,2)}]\dots[P_{(m-1,m)}][Q_{(1,2)}]\dots[Q_{(m-1,m)}][AC_1]\dots[AC_m]$
PITA-CRI	$[D][AC_1]\dots[AC_m][Q_{(1,2)}]\dots[Q_{(m-1,m)}][P_{(1,2)}]\dots[P_{(m-1,m)}]$
PITA-RCI	$[D][Q_{(1,2)}]\dots[Q_{(m-1,m)}][AC_1]\dots[AC_m][P_{(1,2)}]\dots[P_{(m-1,m)}]$
PITA-RIC	$[D]][Q_{(1,2)}]\dots[Q_{(m-1,m)}][P_{(1,2)}]\dots[P_{(m-1,m)}][AC_1]\dots[AC_m]$

Table 7: The format of different templates.

- **BART** (Lewis et al., 2019): The model only uses a BART and converts the AM into a sequence generative formulation. The format of the target sequence is similar to our prompt format.

- **ST** (Morio et al., 2022): This method employs Longformer (Beltagy et al., 2020) and biaffine function for AM.

A.3 Impact of Different Templates

We explore how different types of prompts affect the performance in this section, as shown in Figure 4. We first compare two template variants with our method: a template (denoted by PITA-S) with the same placeholder for different ACs (AC pairs) in each subtask, a template (denoted by PITA-M) with each AC and AC pair following an added *[MASK]* for prediction to simulate the pretrained objective of BART. In addition, We argue that there are task order biases in our PITA model due to the autoregressive generative paradigm. Thus, we investigate the effect of task order in the template and explore five templates (denoted by PITA-ICR, PITA-IRC, PITA-CRI, PITA-RCI and PITA-RIC) in which the positions of placeholders in the three tasks are swapped. The format of all prompt templates can be viewed in Table 7.

Data	Model	TT (min)	IT (sec)
PE	BART	1.17	15.00
	ST	1.23	4.08
	PITA	1.27	2.54
	-w/o TIG	1.20	2.26
CDCP	BART	0.63	18.63
	ST	0.81	1.75
	PITA	0.73	1.26
	-w/o TIG	0.65	1.05

Table 8: Computational cost in terms of Training Time (TT) per epoch (minutes) and Inference Time (IT) in the test set (second).

A.4 Computational Cost

We investigate the computational cost of baseline methods and our PITA model in training and inference. For a fair comparison, all these models use the same batch size of 4 in training and inference. Table 8 shows the training time and inference time on the PE and CDCP. PITA has a competitive efficiency compared to the SOTA model (i.e., ST) for training, while faster than ST for inference. For example, PITA shows a decrease of 1.54s and 0.49s in inference time for all samples in PE and CDCP, respectively, compared to ST, which verifies the efficiency of our method. In addition, during inference, our PITA is more efficient than the generation-based baseline (i.e., BART) by a factor of 6-18 owing to our task interaction prompt learning, although minimal overhead is added to the training process. Comparing PITA with w/o TIG, the task interaction graph introduces a small amount of extra time (almost 1.8s on PE and 1.2s on CDCP for one epoch) in both the training and inference phases (0.28 seconds on PE and 0.21 seconds on CDCP for all instances), which is acceptable in practice.