Prompt for Extraction? PAIE: Prompting Argument Interaction for Event Argument Extraction

Yubo Ma[1†*], **Zehao Wang**[2†*], **Yixin Cao**[‡3]
Mukai Li[4†], **Meiqi Chen**[5†], **Kun Wang**[4], **Jing Shao**[4]
[1] Nanyang Technological University [2] KU Leuven [3] Singapore Management University
[4] SenseTime Group Limited [5] Peking University
yubo001@e.ntu.edu.sg, zehao.wang@esat.kuleuven.be

Abstract

In this paper, we propose an effective yet efficient model PAIE for both sentence-level and document-level Event Argument Extraction (EAE), which also generalizes well when there is a lack of training data. On the one hand, PAIE utilizes prompt tuning for extractive objectives to take the best advantages of Pre-trained Language Models (PLMs). It introduces two span selectors based on the prompt to select start/end tokens among input texts for each role. On the other hand, it captures argument interactions via multi-role prompts and conducts joint optimization with optimal span assignments via a bipartite matching loss. Also, with flexible prompt design, PAIE can extract multiple arguments with the same role instead of conventional heuristic threshold tuning. We have conducted extensive experiments on three benchmarks, including both sentence- and document-level EAE. The results present promising improvements from PAIE (1.1% and 3.8% F1 gains on average in sentence-level and document-level respectively). Further analysis demonstrates the efficiency, generalization to few-shot settings, and effectiveness of different extractive prompt tuning strategies. We will release our codes upon acceptance.

1 Introduction

Understanding text by identifying the event and arguments has been a long standing goal in Natural Language Processing (NLP) (Sundheim, 1992). As shown in Fig. 1, we can quickly understand that the document is talking about a *Sell* event, with four involved arguments, i.e., *Vivendi* (Seller), *Universal Studios* (Artifact), *parks* (Artifact), and *company* (Artifact), where the argument role is in brackets. Since event detection has achieved great success in recent years (Wang et al., 2021), the main challenge lies in Event Argument Extraction (EAE).

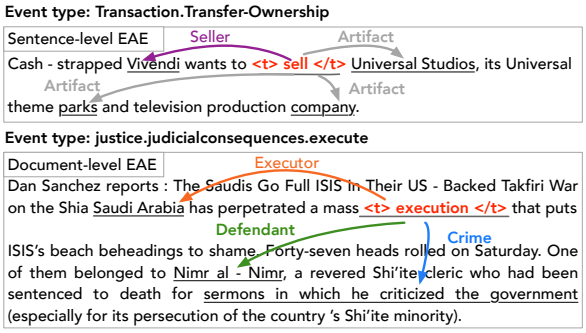

Figure 1: Examples of (top) sentence-level and (bottom) document-level event argument extraction. Trigger words are included in special tokens <t> and </t>. Underlined words denote arguments and arcs denote roles.

Typical efforts in EAE can be roughly classified into two groups. The first group of methods formulates it as a semantic role labeling problem (Wei et al., 2021). There are generally two steps — first identifying candidate spans and then classifying their roles. Although joint models are proposed to optimize them together, high dependence on candidates may still suffer from error propagation (Li et al., 2013). In the second group, recent studies tend to follow the success of Pre-trained Language Models (PLMs) and solve EAE by Question Answering (QA)/Machine Reading Comprehension (MRC) (Liu et al., 2021a; Wei et al., 2021; Du and Cardie, 2020; Liu et al., 2020; Li et al., 2020) and Text Generation (Lu et al., 2021; Li et al., 2021). QA/MRC-based models can effectively recognize the boundaries of arguments with role-specific questions, while the prediction has to be one by one. Generation-based methods are efficient for generating all arguments, but sequential predictions degrade the performance on long-distance and more arguments. Besides, the state-of-the-art performance is still unsatisfactory (around 68% F1 on the widely used dataset ACE05 (Doddington et al., 2004)). Here raise an interesting question, is

[*] Equal Contribution.
[†] Work was done when Yubo, Zehao, Mukai and Meiqi were intern researchers at the SenseTime Group Limited.
[‡] Corresponding Author.

there any way to combine the merits of the above methods, as well as to boost the performance?

This paper targets real scenarios, which require the EAE model to be effective yet efficient at both sentence and document levels, and even under the few-shot setting without sufficient training data. To do this, we highlight the following questions:

- How can we extract all arguments simultaneously for efficiency?
- How to effectively capture argument interactions for long text, without knowing them in advance?
- How can we elicit more knowledge from PLMs to lower the needs of annotation?

In this paper, we investigate prompt tuning under an extractive setting and propose a novel method **PAIE** that **P**rompting **A**rgument **I**nteractions for **E**AE. It extends QA-based models to handle multiple argument extraction and meanwhile takes the best advantage of PLMs. The basic idea is to design suitable templates to prompt all argument roles for PLMs, and obtain role-specific queries to jointly select optimal spans from the text. Thus, instead of unavailable arguments, each role in the template serves as a slot for interactions, and during learning, PLMs tend to fill these slots with exact arguments via a matching loss. By predicting arguments together, PAIE enjoys an efficient and effective learning procedure. Besides, the inter-event knowledge transfer between similar role prompts alleviates the heavy burden of annotation cost.

Specifically, for prompting extraction, we design two span selectors based on role prompts, which select start/end tokens among input texts. We explore three types of prompts: manual template, concatenation template, and soft prompt. They perform well at both sentence-level EAE (S-EAE) and document-level EAE (D-EAE) and ease the requirements of the exhaustive prompt design. For joint span selection, we design a bipartite matching loss that makes the least-cost match between predictions and ground truth so that each argument will find the optimal role prompt. It can also deal with multiple arguments with the same role via flexible role prompts instead of heuristic threshold tuning. We summarize our contributions as follow:

- We propose a novel model, PAIE, that is effective and efficient for S-EAE and D-EAE, and robust to the few-shot setting.
- We formulate and investigate prompt tuning under extractive settings, with a joint selection scheme for optimal span assignments.
- We have conducted extensive experiments on three benchmarks. The results show a promising improvements with PAIE (1.1% and 3.8% F1 gains on average absolutely in S-EAE and D-EAE). Further ablation study demonstrates the efficiency and generalization to few-shot settings of our proposed model, as well as the effectiveness of prompt tuning for extraction.

2 Related Works

Event Argument Extraction: Event Argument Extraction is a challenging sub-task of event extraction (EE). There have been great numbers of studies on EAE tasks since an early stage (Chen et al., 2015; Nguyen et al., 2016; Huang et al., 2018; Yang et al., 2018; Sha et al., 2018; Zheng et al., 2019). Huang and Peng (2021) propose to leverage Deep Value Networks (DVN) that captures cross-event dependencies for EE. Huang and Jia (2021) convert documents to unweighted graph and use GAT to alleviate the role overlapping issue. A common idea is to first identify argument candidates and then fill each with a specific role via multi-label classification (Lin et al., 2020). To deal with implicit arguments and multiple events, Xu et al. (2021) construct a heterogeneous graph of arguments, while DEFNN (Yang et al., 2021) predict arguments via Parallel Prediction Networks.

A recent trend formulates EAE as an extractive question answering (QA) problem (Du and Cardie, 2020; Liu et al., 2020). This paradigm naturally induces the language knowledge from pre-trained language models by converting EAE tasks to fully-explored reading comprehension tasks via a question template. (Wei et al., 2021) considers the implicit interaction among roles by adding constraint with each other in template, while (Liu et al., 2021a) leverages data augmentation to improve the performance. However, they can only predict roles one by one, which is inefficient and usually leads to sub-optimal performance.

To extract all arguments in a single pass, Lu et al. (2021) take EAE as a sequential generation problem with the help of the pre-trained Encoder-Decoder Transformer architecture, such as BART (Lewis et al., 2020) and T5 (Raffel et al., 2020). Li et al. (2021) target generation model by designing specific templates for each event type. In comparison, we prompt argument interactions to guide PLMs and optimize the multiple argument

detection by designing a bipartite matching loss. This not only improves the understanding of long-distance argument dependencies but also enjoys an efficient procedure via prompt-based learning.

Prompt-based Learning: Prompt-based learning is a new paradigm emerging in the field of pre-trained language models (Liu et al., 2021b). Unlike the pre-training and fine-tuning paradigm, which usually asks for an additional classifier, the prompt-based methods convert the downstream tasks to the form more consistent with the model's pre-training tasks. By finding a mapping from a particular word to a category, a classification task can be treated as a Masked LM task (Schick and Schütze, 2021). Recent work found that a small difference in prompt templates may lead to a huge performance gap. Thus, many works explore automatic template generation (Shin et al., 2020), discrete and continuous representations of prompts (Liu et al., 2021c), multiple prompt slots (Qin and Eisner, 2021), etc.. Different from the above prompt tuning method, our proposed method focuses on extraction tasks and prompts PLM for better span selectors.

3 Methodology

PAIE considers multiple arguments and their interactions to prompt PLMs for joint extraction. Our model, as illustrated in Fig. 2, contains three core components: *prompt creation*, *span selector decoding*, and *span prediction*. In the following sections, we will first formulate prompt for extraction, and describe each component in turn.

3.1 Formulating Prompt for Extraction

Existing prompt-based methods mainly focus on classification and generation tasks. Conventional extraction objectives are converted into a generation task. This brings inefficiency issue that the model has to enumerate all of extraction candidates. For example, (Cui et al., 2021) design the prompt for named entity recognition: *[candidate span] is [entity type/not a] entity*. The models need to fill the first slot with candidate entities, and check the outputs of LM for the second slot for extraction. Can prompt-based methods directly be applied on extraction? since the basic idea is similar with classification/generalization — comparing the slot embeddings with label vocabulary/input tokens. Here, we give a formulation about general extractive prompting method, and then apply it on EAE for case study.

(1) *Prompt Creation*. Given context X and a series of queries $Q = \{q_1, q_2, ..., q_K\}$, we create a joint prompt containing all these queries, where f_{prompt} is the prompt creator.

$$Pt = f_{prompt}(Q)$$

(2) *Prompted Selector Decoding*. Given a PLM $\mathcal{L}$, context X, and prompt Pt, we decode a query-specific (answering) span selector as follows:

$$\theta_{q_k} = h_{\mathcal{L}}(q_k; Pt, X)$$

where q_k is the k-th query in the prompt and $h_{\mathcal{L}}$ is the outputs of PLMs.
(3) *Prompted Span Selection*. To find the optimal span, we design two selectors for the start and end tokens from context:

$$(s, e)_{q_k} = \text{Span-search}[g_{\mathcal{L}}(X; \theta_q)]$$

where $(s, e)_{q_k}$ is the span about k-th query and $g_{\mathcal{L}}$ is the span selector. Clearly, such formulation is better than generative extraction by mainly considering the adjacent constraints of span.

Task Definition We formulate EAE task as a prompt-based span extraction problem on dataset D. Given an instance $(X, t, e, R^{(e)}) \in D$, where X denotes the context, $t \subseteq X$ denotes the trigger word, e denotes the event type and $R^{(e)}$ denotes the set of event-specific role types, we aim to extract a set of span A. Each $a^{(r)} \in A$ is a segmentation of X and represents an argument about $r \in R^{(e)}$.

3.2 Prompt Creation for EAE

We create a set of prompts for each event type e in dataset D. Each prompt contains all roles $r \in R^{(e)}$. For example in Fig.2, given event type e as *negotiate* and $R^{(e)}$ as $\{$*Participant*, *Topic*, *Place*$\}$, the prompt $Pt^{(e)}$ may be defined as follows:

Participant communicated with Participant about topic at Place .

We call the mentions of roles in the prompt as **slot**, and there are four slots underlined in this example (and colored in Fig. 2). Such design allows our model to capture the implicit interactions among different roles.

To avoid threshold tuning for multiple arguments with the same role, the prompt is flexible to use multiple slots for the same role, such as role *Participant* in the above example. The number of slots for the role is heuristically determined according to the maximum number of arguments of each role in

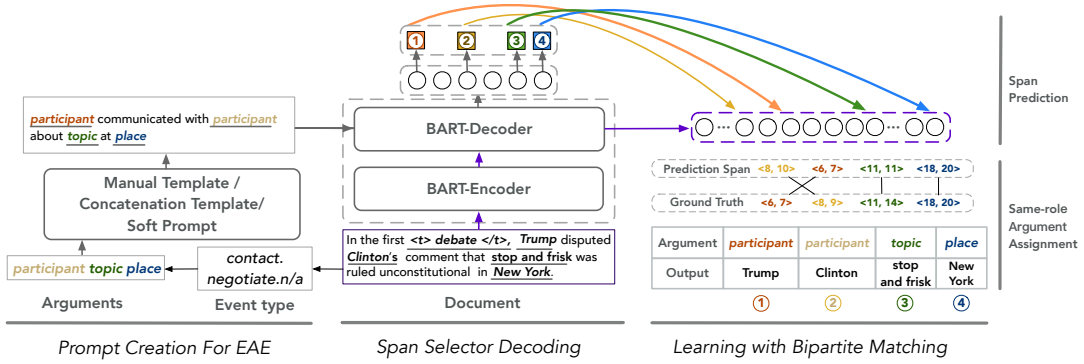

Figure 2: Overall architecture of PAIE. Given a context (about an event), PAIE first creates joint prompt based on its event type. Then the context and prompt are fed into the BART-encoder and BART-decoder to generate context representation and role-specific span selectors. Multiple span selectors extract argument spans from the context simultaneously. A bipartite matching loss finally optimizes the global span assignment.

training dataset. We design three different prompt creators f_{prompt}, the mapping from a set of roles to a prompt as follows:

1. Manual Template: All roles are connected manually with natural language. In experiments, we follow the template from Li et al. (2021) for fair comparison. For multi-argument cases, we simply add slots within brackets as shown in Table 1. The same below.
2. Concatenation Template: To concatenate all role names belonging to one event type.
3. Soft Prompt: Following Qin and Eisner (2021); Liu et al. (2021c), we connect different roles with learnable, **role-specific** pseudo tokens.

We give one example of these three types of prompt in Table 1 and list more examples in Appendix A.6. Further analysis can be found in Section 5.2.

3.3 Span Selector Decoding

Given context X and prompt Pt, this module generates the role-specific span selector θ_k, for each slot k of the prompt. Here we choose pre-trained language model $\mathcal{L}$ as BART (Lewis et al., 2020).

We first define text markers as $\langle$**t**$\rangle$/$\langle$**/t**$\rangle$ then insert them into context X before and after the trigger word respectively.

$$\tilde{X} = [x_1, x_2, ..., \langle\mathbf{t}\rangle, x_{trig}, \langle\mathbf{/t}\rangle, ..., x_n]$$

Instead of concatenating the processed context $\tilde{X}$ and prompt Pt directly, we feed the context into BART-Encoder and the prompt into BART-Decoder separately, as illustrated in Fig. 2. The prompt and context would interact with each other at the cross-attention layers in decoder module.

$$\begin{aligned} H_X^{(enc)} &= \text{BART-Encoder}(\tilde{X}) \\ H_X &= \text{BART-Decoder}(H_X^{(enc)}; H_X^{(enc)}) \\ H_{pt} &= \text{BART-Decoder}(Pt; H_X^{(enc)}) \end{aligned} \quad (1)$$

where H_X denotes the event-oriented context representation and H_{pt} denotes context-oriented prompt representation. For k-th slot in the joint prompt we mean-pool its corresponding representations from h_{pt} and obtain role feature $\psi_k \in R^h$, where h denotes the dimension of hidden layer in BART. Note that a role may have multiple slots and correspondingly, multiple role features and span selectors.

We adopt a simple but effective modification on previous methods by deriving **role-specific span selector** θ_k from every role feature in the prompt. Given role feature ψ_k, we have:

$$\begin{aligned} \psi_k^{(start)} &= \psi_k \circ w^{(start)} \in R^h \\ \psi_k^{(end)} &= \psi_k \circ w^{(end)} \in R^h \end{aligned} \quad (2)$$

where $\theta = [w^{(start)}; w^{(end)}] \in R^{h\times 2}$ is learnable parameters shared among all roles, and $\circ$ represents element-wise multiplication. $\theta_k = [\psi_k^{(start)}; \psi_k^{(end)}]$ is exactly the span selector for k-th slot in the prompt. With only one meta-head θ and simple operations, our method enables to generate arbitrary number of role-specific span selectors to extract related arguments from context. Recall the generation process of role feature ψ_k from prompt h_{pt}, it is obvious that both the interaction among different roles and the information aggregation between context and roles are considered under this paradigm.

Prompt Type	Prompt Example
MA Template	Victor (and Victor) defeated in ConflictOrElection at Place (and Place)
CA Template	Victor (Victor) ConflictOrElection Place (Place)
SF Prompt	<Vic_left0> Victor <Vic_right0> (<Vic_left0> Victor <Vic_right0>) Defeated <Conf_left0> ConflictOrElection <Conf_right0> <Place_left0> Place <Place_right0> (<Place_left0> Place <Place_right0>)

Table 1: Variants of prompt introduced in section 3.2. **MA**:Manual. **CA**:Concatenation. **SF**:Soft. Words with angle brackets in Soft Prompt denote role-specific pseudo tokens of continuous prompt.

3.4 Learning with Bipartite Matching

Bipartite matching aims to find the optimal span assignments for all arguments with the least-cost match. It considers two aspects: argument-role match and same-role argument match.

Span Prediction This module considers argument-role match and aims to detect multiple argument spans for every role simultaneously.

Given the representation of context H_X and all role-specific selectors $\{\theta_k\}$, we follow the extractive prompt formulation in Section 3.1 to calculate the distribution of each token being selected as the start/end of argument for each role feature.

$$\begin{aligned} \text{logit}_k^{(start)} &= \psi_k^{(start)} H_X \in R^L \\ \text{logit}_k^{(end)} &= \psi_k^{(end)} H_X \in R^L \end{aligned} \tag{3}$$

where $\text{logit}_k^{(start)}$ and $\text{logit}_k^{(end)}$ represent start and end position distributions over the context tokens for each slot k, and L denotes the context length.

Then we apply greedy search on predicted start and end position distributions to select the local optimal span for each role-specific selector.

$$(\hat{s}_k, \hat{e}_k) = \underset{(i,j)\in L^2, i<j}{\arg\max} \ \text{logit}_k^{(start)}(i) + \text{logit}_k^{(end)}(j) \tag{4}$$

Same-role Argument Assignment For multiple arguments with the same role, we insert multiple slots about this role and each slot generates one prediction. It is a canonical bipartite matching problem that matches predictions and ground truth as much as possible. Following Carion et al. (2020); Yang et al. (2021), we use Hungarian algorithm (Kuhn, 1955) and leave the detail about it in Appendix A.4.

After finding the optimal assignment $\hat{\sigma}$, we calculate probabilities where the start/end positions locate:

$$\begin{aligned} p_k^{(start)} &= \text{Softmax}(\text{logit}_{\hat{\sigma}(k)}^{(start)}) \\ p_k^{(end)} &= \text{Softmax}(\text{logit}_{\hat{\sigma}(k)}^{(end)}) \end{aligned} \tag{5}$$

Then we define the loss function of the slot k as:

$$\mathcal{L}_k = -(\log p_k^{(start)}(s_k) + \log p_k^{(end)}(e_k)) \tag{6}$$

where s_k and e_k represent the ground truth of start/end positions of the arguments.

For inference, our model efficiently following Eq.4 to extract arguments of each slot, since at most one span is predicted by each slot in the prompt, which avoids the exhaustive threshold tuning.

4 Experiments

In this section, we explore the following questions:

- Can PAIE better utilize PLMs for joint extraction to boost the performance of S-EAE and D-EAE?
- How do different prompt training strategies affect the results?
- How does PAIE perform in various practical settings, including efficiency and generalization to few-shot, long-distance, and multiple arguments?

4.1 Experimental Setup

Datasets We conduct experiments on three common datasets in Event Argument Extraction task: RAMS (Ebner et al., 2020), WIKIEVENTS (Li et al., 2021) and ACE05 (Doddington et al., 2004). RAMS and WIKIEVENTS are latest document-level EAE benchmarks, while ACE05 is a classical dataset commonly used for sentence-level EAE task. We leave the dataset details in Appendix A.1.

Evaluation Metric We adopt two evaluation metrics. (1) Argument Identification F1 score (Arg-I): an event argument is correctly identified if its offsets and event type match those of any of the argument mentions. (2) Argument Classification F1 score (Arg-C): an event argument is correctly classified if its role type is also correct. For WIKIEVENTS dataset, we follow (Li et al., 2021) and additionally evaluate Argument Head F1 score (Head-C), which only concerns the matching of the head word of an argument.

Model	PLM	ACE05		RAMS		WIKIEVENTS		
		Arg-I	Arg-C	Arg-I	Arg-C	Arg-I	Arg-C	Head-C
FEAE (Wei et al., 2021)	BERT-b	-	-	53.5*	47.4*	-	-	-
DocMRC (Liu et al., 2021a)	BERT-b	-	-	-	45.7*	-	43.3*	-
OneIE (Lin et al., 2020)	BERT-b	65.9	59.2	-	-	-	-	-
	BERT-l	73.2	69.3	-	-	-	-	-
EEQA (Du and Cardie, 2020)	BERT-b	68.2*	65.4*	46.4	44.0	54.3	53.2	56.9
	BERT-l	70.5	68.9	48.7	46.7	56.9	54.5	59.3
BART-Gen (Li et al., 2021)	BART-b	59.6	55.0	50.9	44.9	47.5	41.7	44.2
	BART-l	69.9*	66.7*	51.2	47.1	66.8	62.4	65.4
EEQA-BART (Our implementation)	BART-b	68.9	67.0	49.4	46.3	60.3	57.1	61.4
	BART-l	73.1	72.2	51.7	48.7	61.6	57.4	61.3
PAIE (Ours)	BART-b	73.0	70.6	53.0	49.8	68.2	63.4	66.4
	BART-l	**75.7**	**73.3**	**55.6**	**53.0**	**69.6**	**65.7**	**69.2**

Table 2: Overall performance. We highlight the best result and underline the second best. * means the value from the original paper. **b** in column **PLM** denotes base model and **l** denotes large model.

Implementation Details Please refer to Appendix A.3 for implementation details of PAIE.

Baselines We compare PAIE with several state-of-the-art models in three categories: (1) Multi-label classification model: **ONEIE** (Lin et al., 2020) (2) Generation model: **BART-Gen** (Li et al., 2021) (3) QA-based model: **EEQA** (Du and Cardie, 2020), **DocMRC** (Liu et al., 2021a) and **FEAE** (Wei et al., 2021). For fair comparison, we replace the PLMs used in the strongest baseline EEQA with BART, the same with PAIE, namely **EEQA-BART**. More details of baselines are listed in Appendix A.2.

4.2 Overall Performance

Table 2 compares our approach with all baselines. We observe that PAIE performs best on all datasets. For S-EAE, our base model achieves an absolute Arg-C improvement of 3.6%. For D-EAE, our base model obtains 2.4% and 6.3% Arg-C gains on RAMS and WIKIEVENTS, respectively. Similarly, our large-version model achieves 4.3% and 3.3% gains. This demonstrates a good generalization ability of our proposed method on dealing with varying lengths of context.

We also find that QA-based model sometimes performs well even in document-level EAE tasks. The EEQA-BART model shows almost the same Arg-C with BART-Gen (Li et al., 2021) on RAMS dataset. Other QA-based models (especially those considering interactions among arguments, like FEAE (Wei et al., 2021)) also have competitive performance. As for WIKIEVENTS, however, QA-based models are inferior to sequential-generation models significantly. We speculate that the performance of previous QA-based models are not robust to handle longer text. Both BART-Gen (Li et al., 2021) and our model PAIE have a relatively stable performance on various document-level EAE datasets, but our model performs better, especially with smaller PLMs.

Model	Arg-C		
	ACE05	RAMS	WIKI
PAIE	70.6±0.85	49.8 ±0.77	63.4 ±1.04
- bipartite matching	70.3±0.90	49.4±0.69	62.8±0.48
- multi-arg prompt	66.2±1.07	47.7±0.81	61.8±0.85
- role-specific selector	67.0±0.64	46.3±0.77	57.1±0.82
EEQA	65.4	44.0	53.2

Table 3: Ablation study on three benchmarks.

Next, we conduct further analysis with the strongest baseline EEQA-BART and our PAIE. We use the base-version BART for a fair comparison.

4.3 Ablation Study

In this section, we investigate the effectiveness of our main components by removing each module in turn. (1) **bipartite matching**. We drop out the bipartite matching loss and ignore the global optimal span assignment. (2) **multi-arg prompt**. We additionally replace the prompt containing multiple roles with several single templates in which include only one role. (3) **role-specific selector**. The selector is not role-specific anymore but is shared among all roles. This variant degrades to EEQA-BART.

We summarize the results of ablation studies in Table 3. **(1)** EEQA-BART outperforms EEQA significantly, which demonstrates that even conventional QA-based methods have substantial space for improvement with a better PLM and span selection strategy. **(2)** The role-specific selector further im-

Concatenate	PLM	ACE05	RAMS	WIKIEVENTS
✓	BE-b	65.9	46.3	62.9
✓	BA-b	70.2	49.3	62.8
✓	BA-l	72.3	51.7	65.1
✗	BA-b	70.6	49.8	63.4
✗	BA-l	**73.3**	**53.0**	**65.7**

Table 4: Arg-C F1 of different PLMs (BE and BA denote BERT and BART) and usages of prompt (encoder or decoder). Concatenate stands for concatenating prompt with context as inputs of encoder.

proves Arg-C scores in RAMS and WIKIEVENTS, while taking a slightly negative effect on ACE05. Since the former two datasets are document-level and have more role types (65 in RAMS, 59 in WIKIEVENTS, and 36 in ACE05), we speculate that role-specific selector plays a critical role when identifying and disambiguating roles with complicated ontology structures in long documents. **(3)** Joint multi-argument prompt achieves consistent improvement on all three datasets, especially on ACE05 and RAMS. It indicates that the joint prompt has the potential to capture implicit interaction among arguments. **(4)** Bipartite matching loss has an average improvement of 0.4%, and shows a stable optimizing ability due to its permutation-invariance property, which is further discussed in Appendix A.5.

5 Evaluation of Extractive Prompting

5.1 Architecture Variants

Table 4 reports average Arg-C scores of 4 random seeds. We can see that concatenating context and prompt slightly impairs the model performance. It seemingly indicates that the over-interaction between context and prompt is not of benefit. Furthermore, the prompt squeezes the limited input length of the encoder kept for a document if it concatenates with the document. The experiments support our strategy feeding context and prompt separately without concatenation to PAIE.

5.2 Prompt Variants

We investigate how different types of prompts affect the performance, as shown in Fig. 3. We find that **(1)** All three joint prompts outperform the single template (except for the soft prompt on RAMS dataset), which validates the effectiveness of the joint prompt. **(2)** Manual template has the most stable performance and usually the better result than others. **(3)** Concatenation template

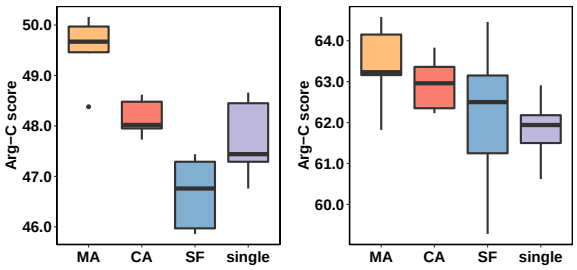

Figure 3: Arg-C F1 using three different types of prompts in Table 1 plus the single template on document-level test set (left) RAMS and (right) WIKIEVENTS.

Model	Trigger-Argument Distance d				
	$-2_{[79]}$	$-1_{[164]}$	$0_{[1811]}$	$1_{[87]}$	$2_{[47]}$
BART-Gen	17.7	16.8	44.8	16.6	9.0
DocMRC	21.0	20.3	46.6	17.2	12.2
FEAE	**23.7**	19.3	49.2	25.0	5.4
EEQA-BART	15.6	24.0	51.7	23.5	8.0
PAIE	21.7	**27.3**	**54.7**	**29.4**	**25.4**

Table 5: Performance (Arg-C F1 score) breakdown by argument-trigger distance d on RAMS development set. The argument number of each case is given in the bracket.

achieves comparable result with manual template. We claim this observation inspiring because the creation of the manual template is laborious and a simple concatenation prompt almost avoids such a handcrafted process. **(4)** A little frustratingly, soft prompt performs relatively poor and unstable, though still slightly better than single template on WIKIEVENTS dataset. It contradicts the current trend of creating distinct continuous prompts which usually perform better than manual ones. We leave this for future work exploring whether there exists competitive continuous prompts in EAE task.

6 Analysis on Real Scenario

6.1 Long-range Dependencies

In D-EAE task, arguments could span multiple sentences. Therefore, the model is required to capture long-range dependencies. For better evaluating PAIE and comparing with others, we list their performance breakdown on different sentence distances between arguments and the given trigger word in Table 5. We can see that PAIE significantly improves the ability to extract arguments with long distances, especially for those behind the given trigger words. We may conclude that PAIE leverages the implicit interaction among roles, and roles conditioning on each other lowers the difficulty to extract long-distance arguments.

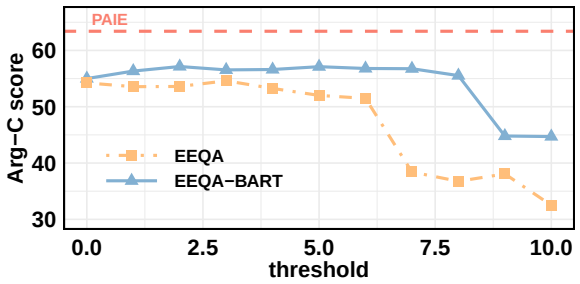

Figure 4: Arg-C F1 w.r.t different thresholds for WIKIEVENTS. We draw the performance of PAIE in red dashed line for comparison (no threshold tuning).

Model	WIKIEVENT Argument Number n			
	$\mathbf{1}_{[468]}$	$\mathbf{2}_{[66]}$	$\mathbf{3}_{[15]}$	$\geq \mathbf{4}_{[17]}$
EEQA-BART	$58.0_{(-6)}$	$59.7_{(-2)}$	$28.6_{(-9)}$	$10.0_{(-18)}$
PAIE (Ours)	**64.7**	**61.4**	**38.1**	**28.6**

Table 6: Arg-C F1 on WIKIEVENTS breakdown by argument number n of one role. The case number is given in the square bracket.

6.2 Same-role Argument Assignment

Multiple arguments may share the same role in the same event. To solve this problem, QA-based methods usually adopt the thresholding strategy, which compares the score of each text span with a manually tuned threshold.

We do a coarse grid search for span threshold on WIKIEVENTS dataset using EEQA model, as shown in Fig. 4. Obviously, the choice of threshold highly affects the performance of the model. In addition, models with the same architecture but different PLMs have totally different optimal thresholds even on the same dataset, not to mention on distinct datasets. Therefore, it consumes lots of time and computational resources for finding a good threshold and usually ends with sub-optimal results.

In PAIE, there is no threshold tuning required since each slot in the prompt only predicts a unique argument span guaranteed by bipartite matching. We evaluate on WIKIEVENTS containing diverse multi-argument cases. Table 6 shows that PAIE outperforms significantly better than QA-based method, especially when dealing with multiple arguments of one role. For roles with three and four or more arguments, PAIE gains an absolute improvement of 9.5% and 18.6%, respectively.

6.3 Few-shot Setting

We analyze how PAIE performs without sufficient annotations on the large-scale RAMS. We also compare with DocMRC, which introduces additional data via data augmentation. Fig. 5 demonstrates superior performance of PAIE, outperforming EEQA-BART and DocMRC performance. Along with the decreasing number of training data, the gains become larger than EEQA-BART. It indicates that PAIE can better utilize PLMs for few-shot settings.

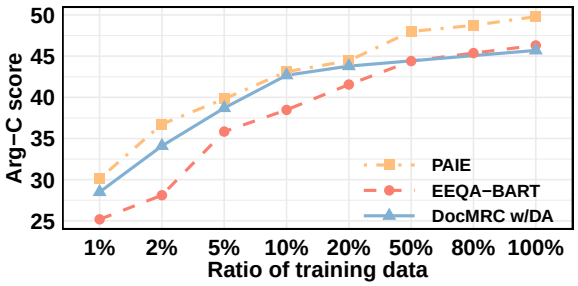

Figure 5: Arg-C F1 score on RAMS test set w.r.t different data ratio. w/DA denotes data augmentation.

Model	ACE05		RAMS		WIKIEVENTS	
	B	L	B	L	B	L
BART-Gen	5.8	12.4	33.2	54.8	19.1	29.0
EEQA-BART	11.8	36.0	66.0	187.4	30.9	83.8
PAIE	**2.9**	**8.4**	**19.0**	**38.6**	**8.4**	**18.3**

Table 7: Inference time in second for different models on the whole test set of ACE05, RAMS, WIKIEVENTS.

6.4 Inference Speed

All previous sections emphasize the superiority of PAIE from the perspective of accuracy performance. PAIE also has much better extraction efficiency compared with other approaches.

In Table 7, we report the overall inference time for different models on single NVIDIA-1080Ti GPU. PAIE runs 3-4 times faster than EEQA, since PAIE predicts multiple roles simultaneously, while EEQA predicts roles one by one. Other QA-based models are likely to have similar speeds with EEQA due to their sequential prediction structure and training process. PAIE is even more advantageous under practical application scenarios since it avoids the heavy threshold tuning.

7 Conclusion

We propose a novel model PAIE that effectively and efficiently extracts arguments at both sentence and document levels. It prompts multiple role knowledge for PLMs for extraction objectives. Also, a bipartite matching loss guarantees the optimal assignment for joint identifying all arguments of the same role. Extensive experiments on three common benchmarks demonstrate our proposed model's effectiveness and the generalization ability in both sentence and document level EAE. We

have also conducted ablation studies on the main components, the extractive prompting strategy, and several real scenarios. In the future, we are interested in investigating co-reference as an auxiliary task of EAE and introducing entity information to better determine argument boundaries.

References

N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. 2020. End-to-end object detection with transformers. In *Proceedings of ECCV*.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 167–176, Beijing, China. Association for Computational Linguistics.

Leyang Cui, Yu Wu, Jian Liu, Sen Yang, and Yue Zhang. 2021. Template-based named entity recognition using BART. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1835–1845, Online. Association for Computational Linguistics.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The automatic content extraction (ACE) program – tasks, data, and evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal. European Language Resources Association (ELRA).

Xinya Du and Claire Cardie. 2020. Event extraction by answering (almost) natural questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 671–683, Online. Association for Computational Linguistics.

Seth Ebner, Patrick Xia, Ryan Culkin, Kyle Rawlins, and Benjamin Van Durme. 2020. Multi-sentence argument linking. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8057–8077, Online. Association for Computational Linguistics.

Kung-Hsiang Huang and Nanyun Peng. 2021. Document-level event extraction with efficient end-to-end learning of cross-event dependencies. In *Proceedings of the Third Workshop on Narrative Understanding*, pages 36–47, Virtual. Association for Computational Linguistics.

Lifu Huang, Heng Ji, Kyunghyun Cho, Ido Dagan, Sebastian Riedel, and Clare Voss. 2018. Zero-shot transfer learning for event extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2160–2170, Melbourne, Australia. Association for Computational Linguistics.

Yusheng Huang and Weijia Jia. 2021. Exploring sentence community for document-level event extraction. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 340–351, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Harold W Kuhn. 1955. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

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

Fayuan Li, Weihua Peng, Yuguang Chen, Quan Wang, Lu Pan, Yajuan Lyu, and Yong Zhu. 2020. Event extraction as multi-turn question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 829–838, Online. Association for Computational Linguistics.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 73–82, Sofia, Bulgaria. Association for Computational Linguistics.

Sha Li, Heng Ji, and Jiawei Han. 2021. Document-level event argument extraction by conditional generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 894–908, Online. Association for Computational Linguistics.

Ying Lin, Heng Ji, Fei Huang, and Lingfei Wu. 2020. A joint neural model for information extraction with global features. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7999–8009, Online. Association for Computational Linguistics.

Jian Liu, Yubo Chen, Kang Liu, Wei Bi, and Xiaojiang Liu. 2020. Event extraction as machine reading comprehension. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1641–1651, Online. Association for Computational Linguistics.

Jian Liu, Yufeng Chen, and Jinan Xu. 2021a. Machine reading comprehension as data augmentation:

A case study on implicit event argument extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2716–2725, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021b. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021c. Gpt understands, too.

Yaojie Lu, Hongyu Lin, Jin Xu, Xianpei Han, Jialong Tang, Annan Li, Le Sun, Meng Liao, and Shaoyi Chen. 2021. Text2Event: Controllable sequence-to-structure generation for end-to-end event extraction. 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 2795–2806, Online. Association for Computational Linguistics.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 300–309, San Diego, California. Association for Computational Linguistics.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying LMs with mixtures of soft prompts. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5203–5212, Online. Association for Computational Linguistics.

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

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269, Online. Association for Computational Linguistics.

Lei Sha, Feng Qian, Baobao Chang, and Zhifang Sui. 2018. Jointly extracting event triggers and arguments by dependency-bridge rnn and tensor-based argument interaction. In *Proceedings of AAAI*.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.

Beth M. Sundheim. 1992. Overview of the fourth Message Understanding Evaluation and Conference. In *Fourth Message Uunderstanding Conference (MUC-4): Proceedings of a Conference Held in McLean, Virginia, June 16-18, 1992*.

Ziqi Wang, Xiaozhi Wang, Xu Han, Yankai Lin, Lei Hou, Zhiyuan Liu, Peng Li, Juanzi Li, and Jie Zhou. 2021. CLEVE: Contrastive Pre-training for Event Extraction. 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 6283–6297, Online. Association for Computational Linguistics.

Kaiwen Wei, Xian Sun, Zequn Zhang, Jingyuan Zhang, Guo Zhi, and Li Jin. 2021. Trigger is not sufficient: Exploiting frame-aware knowledge for implicit event argument extraction. 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 4672–4682, Online. Association for Computational Linguistics.

Runxin Xu, Tianyu Liu, Lei Li, and Baobao Chang. 2021. Document-level event extraction via heterogeneous graph-based interaction model with a tracker. 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 3533–3546, Online. Association for Computational Linguistics.

Hang Yang, Yubo Chen, Kang Liu, Yang Xiao, and Jun Zhao. 2018. DCFEE: A document-level Chinese financial event extraction system based on automatically labeled training data. In *Proceedings of ACL 2018, System Demonstrations*, pages 50–55, Melbourne, Australia. Association for Computational Linguistics.

Hang Yang, Dianbo Sui, Yubo Chen, Kang Liu, Jun Zhao, and Taifeng Wang. 2021. Document-level event extraction via parallel prediction networks. 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

6298–6308, Online. Association for Computational Linguistics.

Shun Zheng, Wei Cao, Wei Xu, and Jiang Bian. 2019. Doc2EDAG: An end-to-end document-level framework for Chinese financial event extraction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 337–346, Hong Kong, China. Association for Computational Linguistics.

A Dataset and Model

A.1 Dataset statistics

We evaluate on three common datasets for Event Argument Extraction: RAMS (Ebner et al., 2020), WIKIEVENTS (Li et al., 2021) and ACE05 (Doddington et al., 2004).

RAMS is a document-level dataset annotated with 139 event types and 65 semantic roles. Each sample is a 5-sentence document, with trigger word indicating pre-defined event type and its argument scattering among the whole document.

WIKIEVENTS is another document-level dataset providing 246 documents, with 50 event types and 59 argument roles. These documents are collected from English Wikipedia articles that describe real-world events and then follow the reference links to crawl related news articles. They also annotate the coreference links of arguments, while we only use the annotations of their conventional arguements in this task.

ACE 2005 is a joint information extraction dataset providing entity, value, time, relation, and event annotation for English, Chinese, and Arabic. We use its event annotation with 33 event types and 35 argument roles for sentence-level EAE tasks. We follow the pre-processing procedure of (Lin et al., 2020), and collect 4859 arguments in the training set, 605 and 576 in the development and test dataset respectively.

Table 8 shows detailed statistics.

Dataset	ACE05	RAMS	WIKIEVENTS
#Sents			
Train	17,172	7,329	5,262
Dev	923	924	378
Test	832	871	492
#Args			
Train	4,859	17,026	4,552
Dev	605	2,188	428
Test	576	2,023	566
#Event	33	139	50
#Role	36	65	59
#Arg per Event	1.19	2.33	1.40

Table 8: Statistics of datasets.

A.2 Details of baseline models

We compare our model with following representative superior models. (1) **ONEIE** (Lin et al., 2020): a joint model extracting entity, relation and event simultaneously. Different from QA-based model, they rely on extracted entities as candidate arguments. (2) **BART-Gen** (Li et al., 2021): a conditional generation model generating (rather than recognizing the spans) arguments sequentially via a sequence-to-sequence model and prompt. (3) **EEQA** (Du and Cardie, 2020): the first Question Answering (QA) based model designed for sentence-level EAE task. (4) **FEAE** (Wei et al., 2021): a QA-based method extended to document-level EAE by considering argument interactions via knowledge distillation. (5) **DocMRC** (Liu et al., 2021a): another QA-based method with implicit knowledge transfer and explicit data augmentation. The implementation details of all baselines are as follow:

1. **FEAE** (Wei et al., 2021): We report the results from the original paper.

2. **DocMRC** (Liu et al., 2021a): We report the results from original paper.

3. **BART-Gen** (Li et al., 2021): For BART-large model, We report the results from origin paper. For BART-base model, we use their code[1] to test its performance on all datasets.

4. **EEQA** (Du and Cardie, 2020): We report the results of ACE05 dataset from the origin papers. We use their code[2] to test its performance on RAMS and WIKIEVENT dataset. In order to generate the question template of these two datasets automatically, we follow the second template setting in **EEQA**. The question temlpate is *What is the ROLE in TRIGGER WORD?*.

5. **EEQA-BART**: For fair comparison with our model, we substitute the pre-trained model of EEQA from BERT to BART and call it as EEQA-BART. We re-train the model on ACE05, RAMS and WIKIEVENT dataset.

[1] https://github.com/raspberryice/gen-arg
[2] https://github.com/xinyadu/eeqa

6. **ONEIE** (Lin et al., 2020): We use their code[3] and re-train the model to get the performance of the model on event argument extraction task (with golden triggers). We don't report its performance on RAMS and WIKIEVENTS because OneIE achieves abnormally low performance on them. Since OneIE is a joint model extracting entity, relation and event, and there is no entity and relation annotation in RAMS and relation annotation in WIKIEVENTS dataset, comparing OneIE with other models is unfair to some extent.

For all the re-trained models mentioned above, we keep all other hyper-parameters the same with default settings in their original papers and search the learning rate in [1e-5, 2e-5, 3e-5, 5e-5]. We report test set performance for the model that performs the best on the development set.

A.3 PAIE model implementation and training setup

PAIE is an extended version of BART-style encoder-decoder transformer. The optimization procedure for one datum is shown in the pseudo code 1. We use pre-trained BART models to initialize the weights of encoder-decoder in PAIE. We train large models on NVIDIA-V100 and base models on NVIDIA-1080Ti. In each experiment, we train the model with 5 fixed seeds (13, 21, 42, 88, 100) and 4 learning rates (1e-5, 2e-5, 3e-5, 5e-5), and vote for the best learning rate for each seed with the best dev-set Arg-C performance. We report the averaged Arg-C performance on the test set for selected checkpoints. For model variations mentioned in Section 5.1, we only change the input strategy and leave other parts constant. We list other important hyperparameters in Table 9.

A.4 Details of Bipartite Matching loss

We formulate the details of bipartite matching loss in the following.

Let us denote $y_r^k = [(s_0, e_0), ..., (s_n, e_n)]$ as ground truth spans of argument role r for datum k, and $\hat{y}_r^k = [(\hat{s}_0, \hat{e}_0), ..., (\hat{s}_m, \hat{e}_m)]$ as predicted spans, where m is the number of occurrence of argument role r in the corresponding prompt.

With the candidate spans for each argument role, we define the bipartite matching between the candidates and ground truth annotations as finding the lowest cost of a permutation Γ of N elements:

$$\hat{\sigma} = \arg\min_{\sigma \in \Gamma_N} \sum_{i}^{N} L_1((s,e)_i, (\hat{s},\hat{e})_{\sigma(i)}) \quad (7)$$

We introduce classical Hungarian algorithm (Kuhn, 1955) for efficient optimal assignment. In Eq.7, N is chosen to the minimum value between m and n, if length of ground truth spans n is larger than number of candidates n, only the optimally matched gold spans are used for loss calculation. Inversely, we will insert $(0, 0)$ to golden answer set to represent a "no answer" case. The $(0, 0)$ span can penalize the over-confidence of span predictions.

After finding the optimal assignment, we calculate span loss for all paired matches. The cross entropy between the start/end logits and the ground truth span is formulated as:

$$\mathcal{L}_{bi}(y_r^k, \hat{y}_r^k) = \sum_{i=1}^{N} \frac{[-\log \hat{p}^s_{\hat{\sigma}(i)}(s_i) - \log \hat{p}^e_{\hat{\sigma}(i)}(e_i)]}{2} \quad (8)$$

In Eq.8, we show the loss under an argument role r of a datum instance k, $p^s_{\hat{\sigma}(i)}$ stands for an optimally assigned prediction for current gold start span index

Algorithm 1: Training one datum

Input: X, Pt `// Context, Prompt tokens`
Data: $Y = \{r_0 : [[s_0^0, e_0^0], [s_0^1, e_0^1]]\}, \{r_1 : [[s_1^0, e_1^0]]\}$
$H_{enc},\ H \leftarrow \text{BART}(X)$
$\hat{P} \leftarrow \text{BART-Decoder}(Pt, H_{enc})$
$L \leftarrow 0$ `// Initialize datum loss`
foreach *role* ***in*** Y*.keys()* **do**
  Set $\hat{Y}_{role}$ **to** empty list
  foreach EMB_{slot} **in** $\hat{P}$.get_next*(role)* **do**
    $\psi \leftarrow \text{MeanPool}(\text{EMB}_{slot})$
    $\psi^{(s)} \leftarrow \psi \circ \boldsymbol{W}^{(s)}$
    $\psi^{(e)} \leftarrow \psi \circ \boldsymbol{W}^{(e)}$
    $\text{logit}^{(s)} \leftarrow \psi^{(s)} H$ `// cos-sim to H`
    $\text{logit}^{(e)} \leftarrow \psi^{(e)} H$ `// cos-sim to H`
    $\hat{Y}_{role}$.insert(
      $\arg\max_{(i,j)\in L^2, i<j} \text{logit}^{(s)}(i) + \text{logit}^{(e)}(j)$
)
  end
  $Y_{role}, \hat{Y}_{role} \leftarrow \text{Hungarian}(Y_{role}, \hat{Y}_{role})$
  $L \leftarrow L + CrossEntropy(Y_{role}, \hat{Y}_{role})$
end

[3] http://blender.cs.illinois.edu/software/oneie/

Hyperparameter	Value
Batch size	24 (ACE05) / 4 (RAMS, WIKIEVENTS)
Weight decay	0.01
Training steps	10000 (ACE05) / 20000 (RAMS, WIKIEVENTS)
Optimizer	AdamW
Adam ϵ	1×10^{-8}
Adam β_1/β_2	0.9 / 0.999
Scheduler	Linear (with 0.1 warmup step)
Maximum gradient norm	5.0
Maxinum encoder sequence length	192 (ACE) / 500 (RAMS, WIKIEVENTS)
Maxinum decoder sequence length	64

Table 9: Hyperparameters for PAIE

Example	**w/ Bipartite**	**w/o Bipartite**
"We demand that the Security Council ... ," said a spokesman for a meeting (*Contact.Meet*) Saturday of **Saddam** and top - level **officials** , quoted by media.	**Entity:** Saddam **Entity:** officials	**Entity:** Saddam **Entity:** ∅
..., bombing at the world-renowned race, where **he** and his brother, **Tamerlan**, 26, set off (*Conflict.Attack.DetonateExplode*) two pressure-cooker bombs near...	**Attacker:**Tamerlan **Attacker:**he	**Attacker:**Tamerlan **Attacker:**∅

Table 10: Examples from our benchmark datasets. Prediction results for models with/without bipartite matching loss. Argument roles are boldfaced in example sentences, trigger word of the event is underlined.

shuffle	**bl**	**ACE**	**RAMS**	**WIKI**
✓	✗	$69.7_{(-0.9)}$	$48.3_{(-1.5)}$	$62.7_{(-0.7)}$
✓	✓	$70.3_{(-0.3)}$	$49.1_{(-0.7)}$	$63.4_{(-0.0)}$
✗	✗	$70.3_{(-0.3)}$	$49.4_{(-0.4)}$	$62.8_{(-0.6)}$
✗	✓	70.6	49.8	63.4

Table 11: An ablation experiment on bipartite loss (bl). Evaluated on all three dataset. Shuffle indicates whether a random shuffle is applied on the order of role annotations in the training set

s_i. Then follow the equation describe the total loss of the model:

$$\mathcal{L}(y, \hat{y}) = \sum_k \sum_r \mathcal{L}_{bi}(y_r^k, \hat{y}_r^k) \tag{9}$$

The full model can be optimized in an end-to-end manner. The bipartite matching is only applied in training. For inference, the model will output all non-zero spans with corresponding argument role as predictions.

A.5 Further analysis of Bipartite Matching

To discuss the effectiveness of bipartite matching, we further go through the annotations in the dataset. Although the task does not guarantee the order of annotations under the same argument role while training, we find the annotators prefer to annotate them in ascending order. This introduces additional annotation bias, which can be captured by model i.e. a late extraction entry in our joint prompt only needs to extract a later annotated span. The modeling towards subordinate relation between same-role arguments in the prompt is downgraded to extracting by their position order. Thus, for a complete analysis, besides an ablation study on the standard training set, we also train on the set in which the order of argument annotations are shuffled. This shuffling process only appears once before the training start with a unique seed for a fair evaluation.

As shown in Table 11, with the standard train set, there is an average drop of 0.4% on Arg-C. With a pre-shuffled train set, the average drop extends to 1%, but at the same time, there is only minimal drop for the model with bipartite loss (second row). PAIE model shows its robustness on permutation. If we further look through the error cases in Table 10, the weakness of multi-argument extraction becomes a typical failure case for the no-bipartite model.

We expected to observe the violation of unique matching towards target span (homogeneous prediction) for the no-bipartite model, especially when permuting role orders. But we did not see cases like that, and this surprises us with the effectiveness of positional embedding for distinguishing tokens even with the same id and unstable loss.

In addition, existing datasets are not designed for evaluating N-to-1 problems, and the cases only appear 8.9% in ACE05, 6.1% in RAMS, 10.9% in WIKIEVENT. The ambiguity of annotations could further reduce the number of effective training data. The importance of bipartite matching in the argument extraction tasks cannot be sufficiently ver-

Prompt Type	Prompt Example
Question Answering Prompt (Du and Cardie, 2020)	Who is the Victor in the Conflict.defeat event? What is the ConflictOrElection in the Conflict.defeat event? Where is the Place in the Conflict.defeat event?
Conditional Generation Prompt (Li et al., 2021)	<arg1> defeated <arg2> conflict at <arg3> place
Our manual template	Victor (and Victor) defeated in ConflictOrElection at Place (and Place)
Our concatenation template	Victor (Victor) ConflictOrElection Place (Place)
Our soft Prompt	<Vic_left0> Victor <Vic_right0> (<Vic_left0> Victor <Vic_right0>) Defeated <Conf_left0> ConflictOrElection <Conf_right0> <Place_left0> Place <Place_right0> (<Place_left0> Place <Place_right0>)

Table 12: Example prompts about Event type *Conflict.Defeat.Unspecified* in WikiEVENTS dataset. Placeholder in Conditional Generation Prompt denotes the content to be filled during decoding stage. Underlined words denote role slots and brackets denote roles with multiple arguments.

ified (no significant performance gap observed). The multi-argument cases in the dataset are almost coordinated, it is a lack of data for evaluating the effectiveness of modeling subordinate relationships between same-role arguments. We expect a large-scale dataset in the future for this purpose.

A.6 Prompt Examples

We first compare our prompt with others used in EAE task in Table 12. The first row gives a standard QA-based prompt, which the model the extraction task as a question answering and expects a question template that can prompt knowledge from PLM, especially for those pretrained on question answering tasks. The second row shows a standard description template for event extraction task, it is usually defined in meta file, works such as (Li et al., 2021) modified the definition with exchangeable placeholders for augment the prompts. Row 3-5 show our three types of prompt respectively. We also show 10 manual template examples about each dataset at Table 13, and the complete version of the prompt would be published with codes later.

B Error Analysis

In this section, We analysis the remaining errors types. We manually check 100 wrong predictions of RAMS dataset and show the distribution in Fig. 6. We only discuss the main categories with examples here.

Annotation Error and Ambiguity. We find that about 27% of the errors are caused by the annotation problem in RAMS dataset. The annotation issues usually contain wrong labeling, missing annotations, and ambiguity of concept. The third issue usually comes from the fungibility of a concept. For instance, “Washington" and “ the United States" can be treated to represent the same political concept in a **contact.collaborate.meet** or **contact.discussion.meet** event. In RAMS, only one of them is annotated and the model may predict the other one.

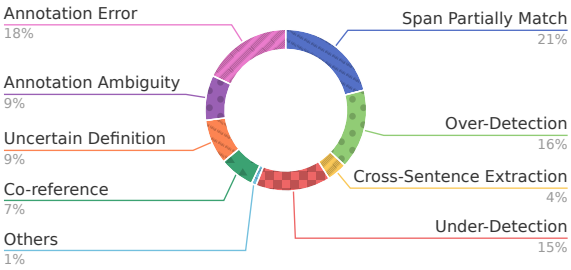

Figure 6: Distribution of error types

Uncertain Definition. This indicates the uncertainty of whether to include a potential entity of an event. For example, in the following sentences:

> “The Syrian government stressed ... and “preventing these organisations from strengthening their capabilities or changing their positions”, in order to avoid **wrecking** the agreement."

a **artifactexistence.damagedestroy.n/a** event has not happened but has potential to do so. The potential role **damagerdestroyer** is mentioned and the model tends to extract it, but this is not annotated in the gold annotations.

Co-reference In multi-sentence-level argument extraction, pronouns are usually used for co-reference. For example, in the following sentences:

> “... Patients don’t feel great, but they’re not **sick** enough to stay home in bed or to be hospitalized ..."

Our model predicts “they" as an answer for the argument role **victim**, though “them" is a reference of the gold annotation “Patients", it is considered as an error according to the current evaluation protocol.

Span Partially Match We find that 21% of the error cases are the partial matching of a span. A large percentage of them fall on the matching of concept but mismatching of span, such as “the center" versus “center". Besides, we also include the cases of multiple correct answers in this category. The correct text can appear in multiple positions in the sentence. The model does not output the exact span but an alternative one. This issue can be mitigated by evaluating on a normalized text (Rajpurkar et al., 2018), this evaluation metric can boost the performance for at least 3% in the benchmarks. Another partial matching happens to an incorrect concept understanding. We find cases for a long human name (contain “-" or “,"), the model only extracts the part before “,". This can be mitigated by introducing stronger entity recognition prior.

Wrong Prediction The wrong prediction of our model mainly contains three categories:

- Over-Detection, which indicates that there is actually no answer contained in the sentences, but our model tends to output a specious result. For example, for the query of some "place" argument, our model finds words representing places that appear in the sentences, even if it does not refer to the place of the current event.
- Under-Detection, which indicates that there is an answer contained in the sentences, but our model fails to extract the related arguments and outputs "No Answer". Some rare arguements only appears in a small amount of training data, which leads the model to give “No Answer” as the ouput.
- Cross-Sentence Extraction, which indicates that the augment extraction needs cross-sentence reasoning, e.g., some entities are mentioned in multiple sentences, the interaction between them needs to be further modeled.

Dataset	Event Type	Natural Lanugage Prompt
ACE05	Movement.Transport	Agent (and Agent) transported Artifact (and Artifact) in Vehicle (and Vehicle) cost Price from Origin place (and Origin place) to Destination place (and Destination place)
	Justice.Arrest-Jail	Agent (and Agent) arrested Person (and Person) at Place (and Place) for Crime
	Justice.Execute	Agent (and Agent) executed Person at Place (and Place) for Crime
	Conflict.Attack	Attacker (and Attacker) attacked Target (and Target) hurting Victims using Instrument (and Instrument) at Place (and Place)
	Contact.Meet	Entity (and Entity) met with Entity (and Entity) at Place (and Place)
	Conflict.Demonstrate	Entity (and Entity) demonstrated at Place (and Place)
	Transaction.Transfer-Ownership	Seller gave Buyer (and Buyer, Buyer, Buyer, Buyer, Buyer, Buyer) to Beneficiary (and Beneficiary, Beneficiary) for the benefit of Artifact (and Artifact, Artifact) cost Price at Place (and Place, Place)
	Transaction.Transfer-Money	Giver (and Giver) gave Money to Recipient (and Recipient) for the benefit of Beneficiary (and Beneficiary) at Place (and Place)
	Life.Be-Born	Person (and Person) was born at Place (and Place)
	Life.Marry	Person married Person at Place (and Place)
RAMS	life.injure. illnessdegradationphysical	Victim person has some physical degradation from Medicalissue imposed by Injurer at Place
	artifactexistence. damagedestroy.destroy	Destroyer destroyed Artifact using Instrument in Place
	conflict.yield.surrender	Surrenderer surrendered to Recipient at Place
	conflict.yield.retreat	Retreater retreated from Origin place to Destination place
	contact.commandorder. correspondence	Communicator communicated remotely with Recipient about Topic at Place
	government.agreements. rejectagreementcontractceasefire	Rejecternullifier rejected or nullified an agreement with Otherparticipant in place
	government.vote. violationspreventvote	Preventer prevented Voter from voting for Candidate on ballot in Place
	inspection.sensoryobserve. physicalinvestigateinspect	Inspector inspected Inspectedentity in Place
	manufacture.artifact. createintellectualproperty	Manufacturer manufactured or created or produced Artifact using Instrument at Place
	life.injure. illnessdegredationsickness	Victim has disease sickness or illness at Place, deliberately infected by Injurer
WIKI-EVENT	ArtifactExistence. ManufactureAssemble	ManufacturerAssembler (and ManufacturerAssembler) manufactured or assembled or produced Artifact (and Artifact) from Components (and Components) using Instrument (and Instrument) at Place (and Place)
	Conflict.Demonstrate	Demonstrator was in a demonstration for Topic with VisualDisplay against Target at Place, with potential involvement of Regulator police or military
	Cognitive.Inspection. SensoryObserve	Observer (and Observer) observed ObservedEntity (and ObservedEntity) using Instrument (and Instrument) in Place (and Place)
	Cognitive. TeachingTrainingLearning	TeacherTrainer (and TeacherTrainer) taught FieldOfKnowledge (and FieldOfKnowledge) to Learner (and Learner) using Means (and Means) at Institution (and Institution) in Place (and Place)
	Control.ImpedeInterfereWith	Impeder (and Impeder) impeded or interfered with ImpededEvent at Place (and Place)
	Transaction.Donation	Giver gave ArtifactMoney to Recipient (and Recipient) for the benefit of Beneficiary (and Beneficiary) at Place (and Place)
	Disaster.DiseaseOutbreak	Disease (and Disease) broke out among Victim (and Victim) or population at Place (and Place)
	Justice.TrialHearing	Prosecutor tried Defendant (and Defendant) before JudgeCourt for Crime (and Crime) in Place (and Place)
	Medical.Vaccinate	Treater vaccinated Patient via VaccineMethod for VaccineTarget at Place (and Place)
	Personnel.StartPosition	Employee started working in Position at PlaceOfEmployment organization in Place (and Place)

Table 13: Example prompts. Underlined words denote role slots and brackets denote N-to-1 roles.