Data-Scarce Event Argument Extraction: A Dynamic Modular Prompt Tuning Model Based on Slot Transfer

Anonymous ACL submission

Abstract

Event Argument Extraction (EAE) facilitates comprehension of the text related to an event by extracting and analyzing event arguments. The superiority of previous studies typically rises from the abundance of high-quality event annotations, which is labor-intensive to produce and hard to satisfy by reality. In this paper, we approach data-scarce EAE in both low-resource and few-shot scenarios, which have far-reaching implications for practice. Specifically, we propose a model called Dynamic Modular Prompt Tuning based on Slot-Transfer (DAMPT)[1], which dispenses with any manual effort usually required in existing methods. DAMPT turns to large-scale language models to generate dynamic modular prompts, which are more adaptable than the static ones manually given by experts. Furthermore, DAMPT incorporates a prompt-tuning algorithm called slot-transfer to facilitate event-specific knowledge transfer. An extensive experimental evaluation validates the effectiveness and generalization ability of DAMPT in data-scarce scenarios.

1 Introduction

Event extraction is an essential text comprehension task aiming to extract structured event information from unstructured text. The downstream applications of event extraction consist of event relation extraction and knowledge graph construction. As a crucial sub-task of event extraction, Event Argument Extraction (EAE) seeks to determine event roles (such as participants, time, and location of an event) when given trigger words. As shown in Figure 1(a), given a context with the event type *Personnel.Start-Position* whose trigger word **"hired"** is indicated by **'<t>'** and **'</t>'**, EAE aims to extract the arguments of the event, that is, **"tabloid"** (i.e., role Entity), **"peter arnett"** (i.e., role Person), **"correspondent"** (i.e., role Position), and **"baghdad"** (i.e., role Place).

[1]We will publicize our code after the paper has been accepted

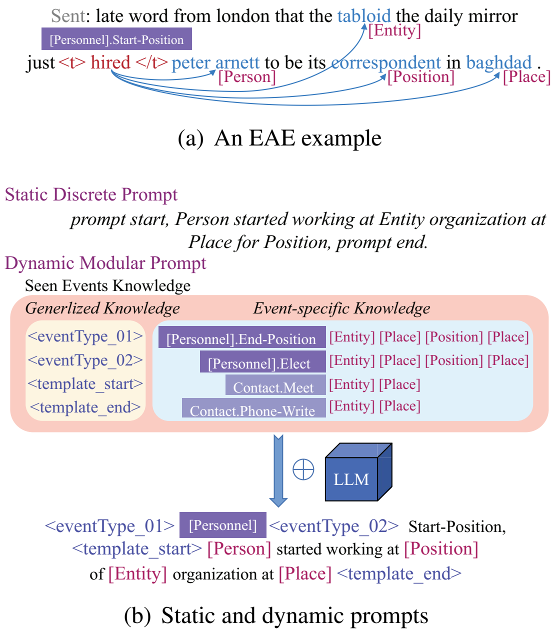

Figure 1: (a) An EAE example. (b) Static and dynamic prompts for the context in (a): compared to the static discrete prompts, the dynamic modular prompts have tunable components and generalized and event-specific knowledge by learning continuous vectors.

EAE learning usually relies on supervised solutions with abundant annotated event arguments (Chen et al., 2015; Nguyen et al., 2016; Yang and Mitchell, 2016). However, data scarcity is a prevalent challenge in the real world, rendering EAE a pragmatic yet challenging approach. One feasible solution for data-scarce EAE is transfer learning (Lyu et al., 2021; Zhang et al., 2021), which can transfer knowledge from other tasks such as Textual Entailment (TE), Question Answering (QA), and Semantic Role Labeling (SRL). However, this strain of approaches depends on the capabilities of the models originally designed for other tasks, and the specific challenges and properties of EAE are not explicitly addressed.

Another solution is Pre-trained Language Model (PLM), which has demonstrated great language understanding ability by inducing prompts for new events (Li et al., 2021; Hsu et al., 2022; Ma et al., 2022). As depicted in Figure 1(b), these prompts are typically event templates designed by experts. While PLM-based methods have shown great promise, they depend on manually crafted discrete prompts[2]. In addition to high costs of human labeling, static discrete prompts offer no opportunity for tuning, making them nearly impossible to transfer event knowledge from insufficient event annotations. To alleviate these restrictions, Liu et al. (2022) introduced dynamic prefix tuning for EAE. This approach still relies on manually designed event description templates and focuses only on tuning type information.

Considering the aforementioned shortcomings of current methodologies, two questions lead us to propose a PLM-based EAE model: (i) *How to induce dynamic prompts that eliminate any manual intervention while maintaining semantic information for a newly emerging event type*, and (ii) *how to utilize the capabilities of PLM to acquire both generalized and event-specific knowledge for EAE.* As portrayed in Figure 1(b), we propose to structure prompts with generalized knowledge (suggesting common information) and event-specific knowledge (describing specific event types and roles), facilitating the tunability of prompts from multiple perspectives.

The proposed model, which we call Dynamic Modular Prompt Tuning based on Slot-Transfer (DAMPT), generates dynamic prompts for new events through an Event Type Module and an Event Template Module, which separately refine the type and role semantic information. To generate event templates contained in Event Template Module without any manual intervention, we utilize Large Language Models (LLMs) through in-context learning. To model generalized knowledge, we introduce continuous module indicator vectors. Additionally, we propose a Slot-Transfer algorithm to model event-specific knowledge, where top-level sub-types and event roles are treated as specific slots. We allow PLMs to transfer knowledge from only a few accessible events to new events by tuning slot representations.

To sum up, our contributions are as follows:

[2]Discrete prompts are actual strings in text, while continuous prompts are extracted from embedding spaces (Liu et al., 2023b).

- We propose DAMPT - the first attempt to automatic prompt construction in EAE without any additional annotations.
- We improve knowledge transfer in both generalized and event-specific prompt representations, which enables dynamic prompts incorporating tunable components.
- Experiments carried out in both low-resource and few-shot settings effectively demonstrate the efficiency and effectiveness of DAMPT in scenarios characterized by data scarcity.

2 Related Work

2.1 PLM-Based EAE

The powerful language ability of PLM enables existing PLM-based methods to achieve high performance. Given a few example data, PLM-based methods can accomplish data-scarce EAE by inducing appropriate prompts. These methods can be roughly split into two types, QA-based methods and generation-based methods.

QA-based methods formulate the EAE task as a question-answering problem wherein prompts are designed as questions asked against events. Du and Cardie (2020) developed a series of steps to generate questions for different event types and roles. Liu et al. (2020) treated EAE as a machine reading comprehension (MRC) problem and built large corpora with manually designed descriptive statements to train a question generation model. Liu et al. (2021) leveraged MRC to generate augmented training data and transferred knowledge using a unified MRC framework.

In contrast, generation-based methods fill event templates with the missing event arguments. Li et al. (2021) constructed prompts by replacing the event role in ontology event templates with placeholders. Hsu et al. (2022) utilized additional weakly supervised information and semantic information of event roles for EAE. Ma et al. (2022) constructed prompts incorporating event templates and treated role representations as selectors to jointly select argument spans. Dai et al. (2022) presented a bi-directional iterative prompt-tuning method. Liu et al., 2022 devised a prefix-tuning strategy in the PLM-based template-filled process. Ren et al. (2023) designed the retrieval strategy for EAE to augment text generation.

2.2 EAE in Data-Scarce Scenarios

Hsu et al. (2022) focuses on low-resource event

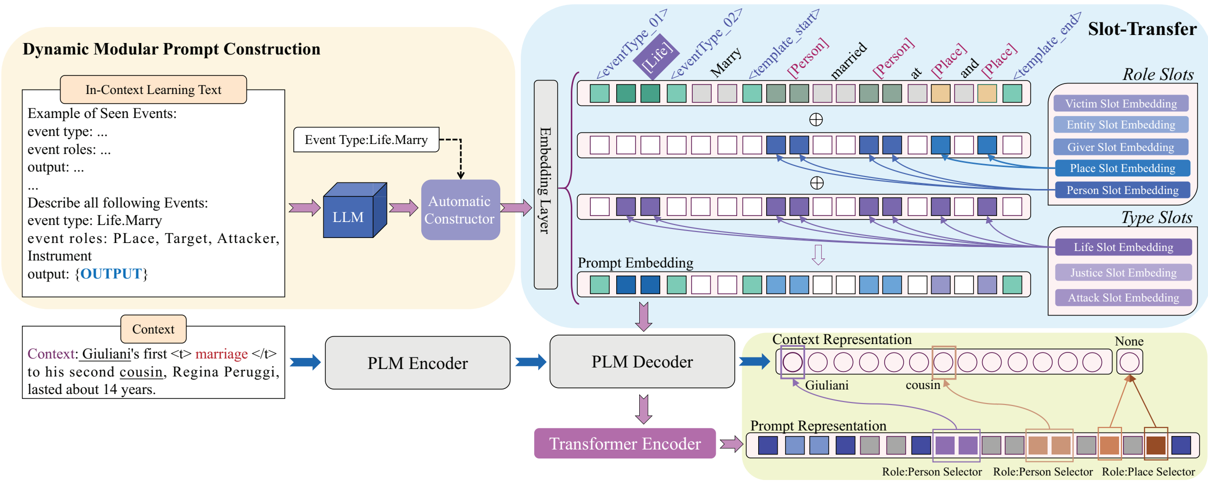

Figure 2: Overview of DAMPT. An example context of type **Life.Marry** is input with its trigger word. Firstly, the context representation is generated through PLM. Next, Dynamic Modular Prompt is automatically constructed from event type and event template generated by LLM. In Decoder Embedding Layer, the Role slots of **Person** and **Place** and top-level Type slot of **Life** are fused with prompt semantic embedding. Finally, after interacting with the context in PLM Decoder, the prompt is fed into Transformer Encoder to capture interactions between argument selectors, which then extract argument spans from context presentation.

extraction and formulates event extraction as a conditional generation problem. Yao et al. (2023) introduces a retrieval-augmented approach for data-efficient knowledge graph construction, dynamically leveraging schema-aware Reference As Prompt. Liu et al. (2023a) employed a chain reasoning paradigm to capture long-range interdependence. Hsu et al. (2023) integrated abstract meaning representation into the model.

EAE in the zero/few-shot setting is a more challenging task. Huang et al. (2018) designed a transferable neural network model that could map event mentions and types into a shared semantic space. Liu et al. (2020) turned to abundant MRC datasets to generate schema-defining questions. Lyu et al. (2021) transferred pre-trained TE/QA models to EAE, while Zhang et al. (2022) performed transfer learning from SRL to EAE. Zhang et al. (2023) leveraged both overlapping knowledge across datasets and dataset-specific knowledge.

2.3 Prompt-Tuning Methods

It is natural to guide PLM using discrete prompts for appropriate text comprehension (Brown et al., 2020; Gao et al., 2021). However, the power of discrete prompts may be restrained during manual construction. To settle this issue, a few studies have proposed prompt tuning methods, which aimed to learn continuous prompts that were integrated into the inputs (Li and Liang, 2021; Hambardzumyan et al., 2021; Zhong et al., 2021).

3 Methodology

An overview of DAMPT is illustrated in Figure 2.

Task Definition: Given a context $c_i \in C$ represented as $c_i = [w_{i,1}, w_{i,2}, ..., \langle t \rangle, v_i, \langle /t \rangle, ..., w_{i,L}]$, where the trigger word v_i is surrounded by tokens $\langle t \rangle$ and $\langle /t \rangle$, and its event type $e_j \in E$ with specific event roles $\{r_{j,1}, r_{j,2}, ..., r_{j,k}\}$, the task is to extract argument spans $\{a_{i1}, a_{i2}, ..., a_{ik}\}$ by inducing an appropriate prompt $p_j \in P$. $a_{i,k}$ is represented as $a_{i,k} = \{(m_k, n_k) | m_k, n_k \in (0, L)\}$, where m_k and n_k are the begin and end positions.

3.1 Dynamic Modular Prompt

Dynamic Modular Prompt is devised to induce p_j whose top-level subtype e_j^{top} of event type e_j and event roles $\{r_{j,1}, r_{j,2}, ..., r_{j,k}\}$ are tunable. As shown in Figure 3, Dynamic Modular Prompt consists of two modules, *Event Type Module* and *Event Template Module*.

3.1.1 Event Type Module

The semantics of an event argument is closely related to its event type. Additionally, the same event role may appear in different event types with different semantics. Based on these observations, Event Type Module enriches event-type information for an automatic construction of dynamic prompts.

In particular, module indicator tokens **<eventType_01>** and **<eventType_02>** are embedded for each level of e_j in module[3], where

[3]For the datasets with more levels of event type hierarchies,

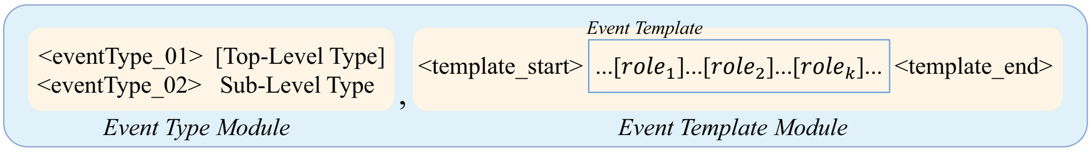

Figure 3: Dynamic Modular Prompt. It consists of the Event Type Module and Event Template Module.

the top-level type represents the general category of event and contains multiple sub-level types. For example, in the event type *Life.Die*, *Life* is the top-level type and *Die* is the sub-level type. Under the top-level *Life*, there are other event types such as *Life.Injure*, *Life.Marry*, *Life.Born*, etc. Thus, the top-level types embody the coarse-grained type information that shared among events. As a result, we treat the top-level types as slots in Section 3.2.

3.1.2 Event Template Module

An event template is a natural language sentence used to describe an event type containing its roles. As indicated in Figure 3, an event template is surrounded by two indicator tokens **<template_start>** and **<template_end>**.

Existing models use event templates obtained from the dataset ontology created by experts. On the contrary, we use in-context learning with LLMs (such as GPT-3.5 [4]) to automatically generate event templates without any human effort.

GPT-generated template We utilize the large-scale model GPT-3.5 for template generation due to its strong in-context learning capability (Brown et al., 2020). Specifically, we feed it with a few event templates to generate new templates for all event types. The example section in the context given to GPT-3.5 includes event type names, event role names, and the corresponding templates. In the output section, we guide the generation of output with event type and event role names.

Eventually, the dynamic modular prompt p_j for the event type e_j is constructed as

$$p_j = Concate(M_1(e_j), M_2(e_j, r_{j,1}, r_{j,2}, ..., r_{j,k})), \quad (1)$$

where M_1 and M_2 respectively denote Event Type Module and Event Template Module, $Concate(\cdot,\cdot)$ represents concatenation of two outputs.

3.2 Prompt Tuning with Slot-Transfer

To improve the prompt p_j with event-specific knowledge, we propose a prompt tuning method, called *Slot-Transfer*. Continuous event *Role Slots* and event *Type Slots* with Dynamic Modular Prompt help to transfer event role and type knowledge. Finally, we fuse Role Slot embedding and Type Slot embedding with the semantic embedding of p_j.

3.2.1 Role Slot Transfer

To incorporate event-specific knowledge beyond the semantic information related to event roles, we propose to transfer event role knowledge by tuning Role Slots in the prompt. In our Dynamic Modular Prompt, we fuse Role Slots at the position of each role in $\{r_{j,1}, r_{j,2}, ..., r_{j,k}\}$. For each specific Role Slot, we derive a particular slot embedding, and each event role shares its slot embedding across all event types in which it appears.

We first embed the semantic information of p_j as follows:

$$\rho_j = DecEmd(p_j) = \{t_1, t_2, ..., t_z\}, \quad (2)$$

where $DecEmd(\cdot)$ represents the embedding layer of PLM decoder. Then, for the event role $r_{j,k}$ in ρ_j, we fuse its specific Role Slot embedding with its semantic embedding via a gate vector $g_{j,k}$ as follows:

$$\begin{aligned} idx &= Index(r_{j,k}), \\ g_{j,k} &= Sigmoid(W_1 t_{idx} + W_2 R_{j,k} + b_1), \quad (3) \\ t'_{idx} &= g_{j,k} \odot t_{idx} + (1 - g_{j,k}) \odot R_{j,k}, \end{aligned}$$

where $Index(\cdot)$ returns the index of $r_{j,k}$ in ρ_j and W_1 and W_2 are learnable parameters. t_{idx} is the token embedding of $r_{j,k}$ in ρ_j obtained in Eq. (2), and $R_{j,k}$ is the specific Role Slot embedding of $r_{j,k}$, which is randomly initialized and tuned by PLM.

3.2.2 Top-Level Type Slot Transfer

The Role Slot embedding t'_{idx} is befitting for a specific event type, while an event type can form different contexts for different roles. Additionally, there exist semantic differences for the same event role in different top-level event types. For example, the role *Agent* means "a person whose job is to manage the affairs of other people in business" in

we add more component indicator tokens and treat the levels beyond the last level as slots.

[4] https://openai.com/blog/chatgpt

the *Personnel.Nominate* events; in the *Life.Injure* events, it means "the person or thing that does an action".

Based on the above observations, we introduce top-level Type Slot to enrich the event-specific knowledge of prompts from the type view. In addition to treating top-level event types as Type Slots in the Event Type Module, we also fuse top-level Type Slots with event roles in Event Template Module. Top-level Type Slots can transfer knowledge across different event types, facilitating improving prompts with event type commonality information.

For the top-level event type e_j^{top} in the Event Type Module, whose semantic embedding in ρ_j is γ_j, we fuse the top-level Type Slot embedding $T_{e_j^{top}}$ with γ_j in accordance with the same strategy in Eq. (3). For each event role, we also fuse its top-level Type Slot embedding with its embedding t'_{idx} obtained in Eq. (3). By virtue of incorporating Role Slot embedding and top-level Type Slot embedding, the newly generated prompt ρ'_j can assimilate event-specific knowledge.

3.3 Interaction of Arguments

Event arguments usually interact with others. For instance, we can induce the *Justice.Sue* event from the sentence "he's been <t> sued </t> by an auction house for non-payment, and by a concert promoter for allegedly backing out of two millennium performances". After extracting the argument **"house"** that corresponds to the role *Plaintiff*, we can also capture the argument **"prompter"** which is connected with **"house"** by the conjunction **"and"** and infer it is also a *Plaintiff*. Argument interactions can provide hidden information valuable for data-scarce EAE learning.

In pre-trained large generative models (e.g., BART (Lewis et al., 2020)), although self-attention and cross-attention mechanisms are used to capture interaction information, earlier extractions cannot directly access the information of later extractions. To address this issue, we employ a Transformer Encoder (Vaswani et al., 2017) as a simple Argument Interaction Module after generating argument selectors, which will be formulated in Section 3.4. In Section 4.4: Ablation Studies and Section 4.5: Case Studies we will evaluate the effectiveness of our interaction module.

3.4 Extraction Process

Our extraction approach is inspired by the method proposed by Ma et al. (2022), which treats event role representations generated by PLMs as span selectors and then use the generated selectors to extract argument spans.

We encode the context c_i via PLM Encoder and obtain its hidden state $\widehat{H}_{c_i}$ which serves both as input and as a hidden state of PLM Decoder. Then, we can obtain the context representation H_{c_i} through PLM Decoder. Additionally, we incorporate the prompt embedding ρ'_j mentioned in Section 3.2 as input of PLM Decoder and process cross-attention computation between ρ'_j and $\widehat{H}_{c_i}$, yielding the final prompt representation $H_{\rho'_j}$ as follows:

$$\begin{aligned} \widehat{H}_{c_i} &= Encoder(c_i) \in R^{L\times h}, \\ H_{c_i} &= Decoder(\widehat{H}_{c_i}, \widehat{H}_{c_i}) \in R^{L\times h}, \\ H_{\rho'_j} &= Decoder(\rho'_j, \widehat{H}_{c_i}) \in R^{|\rho'_j|\times h}, \end{aligned} \tag{4}$$

where L is the length of the context c_i and h is the hidden size.

Next, we model the argument interaction mentioned in Section 3.3 by Transformer Encoder:

$$\begin{aligned} Z &= LNorm(H_{\rho'_j} + MultiAtt(H_{\rho'_j})), \\ H'_{\rho'_j} &= LNorm(Z + FF(Z)) \in R^{|\rho'_j|\times h}, \end{aligned} \tag{5}$$

where $LNorm(\cdot)$, $MultiAtt(\cdot)$, and $FF(\cdot)$ separately represent LayerNorm, MultiHead Attention, and FeedForward process in Transformer Encoder.

The mean pooling output $\epsilon_k \in R^h$ of token embeddings which correspond to $r_{j,k}$ in $H'_{\rho'_j}$, is then transformed to the argument start span selector $\varepsilon_k^s \in R^h$ and end span selector $\varepsilon_k^e \in R^h$. Finally, the argument span is extracted as follows:

$$\begin{aligned} \begin{bmatrix} \varepsilon_k^s \\ \varepsilon_k^e \end{bmatrix} &= \begin{bmatrix} \epsilon_k \\ \epsilon_k \end{bmatrix} \odot \begin{bmatrix} w^s \\ w^e \end{bmatrix} \in R^{2h}, \\ \begin{bmatrix} l_k^s & l_k^e \end{bmatrix} &= H_{c_i} \begin{bmatrix} \varepsilon_k^s & \varepsilon_k^e \end{bmatrix} \in R^{2L}, \\ \begin{bmatrix} \hat{y}_k^s & \hat{y}_k^e \end{bmatrix} &= Softmax(\begin{bmatrix} l_k^s & l_k^e \end{bmatrix}) \in R^{2L}, \end{aligned} \tag{6}$$

where w^s and w^e are learnable parameters, and H_{c_i} is the context representation obtained in Eq. (4).

The loss function is defined as

$$\begin{aligned} L_k(c_i) &= -\log \hat{y}_k^s(y_k^s) - \log \hat{y}_k^e(y_k^e), \\ L &= \sum_{c_i \in C} \sum_k L_k(c_i), \end{aligned} \tag{7}$$

where $\hat{y}_k^s$ and $\hat{y}_k^e$ are prediction vectors for the argument start position and end position, and y_k^s and y_k^e are ground truth.

Model	PLM	Arg-I						Arg-C					
		1%	3%	5%	10%	20%	30%	1%	3%	5%	10%	20%	30%
ACE05-E													
DEGREE	BART-b	13.3	17.2	27.1	34.5	49.0	63.9	12.6	16.6	22.4	30.0	43.2	58.6
BART-Gen	BART-b	29.2	31.9	35.3	41.1	48.0	51.0	23.2	28.3	32.5	37.0	41.0	45.8
BERT-QA	BERT-b	30.3†	43.7†	48.9	50.8	55.5	56.8	20.1	39.0†	44.4	48.5	51.8	52.6
PAIE	BART-b	21.9	41.2	49.2	56.5†	63.7	67.6†	19.2	34.4	43.5	51.6	59.3	62.8
DAMPT	BART-b	**31.3**†	**44.9**	52.4†	**59.5**	**64.5**	**67.9**	**27.4**	**39.1**	47.4†	**53.5**	**60.7**	63.3†
w/o GPT-Gen	BART-b	28.3	43.0	**53.5**	55.9	64.3†	66.9	26.0†	**39.1**	**47.7**	52.8†	60.1†	**63.5**
WIKIEVENTS													
BART-gen	BART-b	–	14.2	15.0	31.8	35.9	51.2	–	11.4	13.5	28.5	30.9	47.5
PAIE	BART-b	**37.7**	52.4†	53.8	62.8†	67.9†	63.7	**32.4**	47.7	48.4	56.7†	61.8†	59.9
DAMPT	BART-b	36.3†	51.4	54.1†	61.5	**68.8**	**67.5**	31.6†	48.1†	49.6†	56.4	**62.1**	**62.0**
w/o GPT-Gen	BART-b	35.7	**52.5**	**56.0**	**63.5**	65.0	66.5†	30.6	**48.5**	**51.1**	**58.4**	60.4	61.4†

Table 1: Low-resource EAE performance of the proposed DAMPT model and the baselines. w/o GPT-Gen is the variant of DAMPT that uses human-designed templates as other models. **Bold** score is the best, and the symbol † indicates the second-best.

4 Experiments

We conduct EAE experiments on low-resource and few-shot scenarios to analyze the performance of DAMPT. The implementation details are delineated in Appendix A.

4.1 Experimental Settings

Datasets We conduct experiments on the sentence-level EAE dataset ACE05-E (Doddington et al., 2004) as well as the document-level EAE dataset WIKIEVENTS (Li et al., 2021), following the pre-processing steps outlined in Ma et al. (2022). The statistics of each dataset are shown in Table 5 in Appendix A due to space limitation.

Data Splits for Low-Resource and Few-Shot Settings For low-resource setting, we generates different proportions (1%, 3%, 5%, 10%, 20%, 30%) of ACE05-E and WIKIEVENTS training data as the same as Hsu et al. (2022). In addition, we perform the same zero-shot split on the ACE05-E dataset as Huang et al. (2018), that is, the top-10 frequent event types in the train set can be seen, while all of the 23 event types in the test set are unseen. As to the few-shot scenario, we respectively take 5-shot setting and 10-shot setting, where 5 and 10 samples of each unseen-type event are let into the train set. For the WIKIEVENTS dataset, we have 10 seen types and 40 unseen types.

Evaluation Metrics We adopt Argument Identification F1 score (Arg-I), Argument Classification Precision (P), Recall (R) and F1 score (Arg-C) as evaluation measures (Ma et al., 2022, Hsu et al., 2022). These Arg-based criteria are strict since they deem an argument as correctly classified only when its span, event type, and role type all match the corresponding ground truth. Achieving a high Arg-based score indicates relatively comprehensive success in EAE.

Baselines We compare DAMPT with the following state-of-the-art models: **(1) classification-based models**: OneIE (Lin et al., 2020) and TSAR (Xu et al., 2022), **(2) generation-based model**: BART-gen (Li et al., 2021), DEGREE (Hsu et al., 2022), **(3) QA-based model**: BERT-QA (Du and Cardie, 2020), and **(4) prompt-based models**: PAIE (Ma et al., 2022) and BIP (Dai et al., 2022).

4.2 Main Results

In Table 1, we compare the performance of our model and the baselines on ACE05-E and WIKIEVENTS in low-resource settings. To provide a detailed illustration, we also report the performance of DAMPT (w/o GPT-Gen), which excludes the impact of event templates generated by GPT. It can be observed that DAMPT outperforms all of the baselines on different proportions of training data of ACE05-E. Specifically, DAMPT achieves a 4.2% improvement under Arg-C with the use of only 1% of the ACE05-E training data, which demonstrates the effectiveness of DAMPT in extreme data-scarce scenarios. On WIKIEVENTS, DAMPT performs well when the proportion is larger than 1%, achieving a 2.7% Arg-C improvement with 5% WIKIEVENTS training data.

In summary, DAMPT is superior to or comparable with existing state-of-the-art models on the ACE05-E and WIKIEVENTS datasets.

Model	PLM	zero-shot P	zero-shot R	zero-shot Arg-C	5-shot P	5-shot R	5-shot Arg-C	10-shot P	10-shot R	10-shot Arg-C
ACE05-E										
OneIE	BERT-b	–	–	–	10.3	10.4	10.3	13.4	15.6	14.5
BART-gen	BART-b	53.0	45.6	49.0	57.2	46.9	51.5	62.4	49.7	55.3
BERT-QA	BERT-b	49.5	55.8	52.4	66.3	52.0	58.3	60.6	57.7	59.2
DEGREE	BART-l	–	–	53.3	–	–	61.7	–	–	64.3
BIP	BERT-b	54.6	$60.9^{\dagger}$	57.6	58.6	**66.5**	62.3	60.6	68.3	64.2
PAIE	BART-b	$59.2^{\dagger}$	53.1	56.0	64.9	64.3	64.6	**68.2**	66.3	67.2
DAMPT	BART-b	**60.2**	**64.3**	**62.2**	$67.0^{\dagger}$	65.8	$66.9^{\dagger}$	66.2	$70.0^{\dagger}$	$68.1^{\dagger}$
w/o GPT-Gen	BART-b	**60.2**	57.4	$58.8^{\dagger}$	**68.3**	$66.1^{\dagger}$	**67.2**	$67.8^{\dagger}$	**70.8**	**69.3**
WIKIEVENTS										
BART-gen	BART-b	48.6	42.6	45.4	51.5	47.2	49.3	44.4	37.2	40.5
TSAR	BERT-b	18.2	19.6	18.9	52.8	**61.8**	57.0	53.1	**63.8**	$57.9^{\dagger}$
PAIE	BART-b	50.0	48.8	49.4	55.8	58.5	57.1	**59.7**	54.9	57.2
DAMPT	BART-b	$51.4^{\dagger}$	**52.7**	**51.0**	**63.6**	56.8	**60.0**	$57.2^{\dagger}$	$58.8^{\dagger}$	**58.0**
w/o GPT-Gen	BART-b	**52.2**	$49.5^{\dagger}$	$50.1^{\dagger}$	$57.7^{\dagger}$	$58.8^{\dagger}$	$58.3^{\dagger}$	56.5	57.6	57.1

Table 2: Few-shot EAE performance of the proposed DAMPT model and the baselines.

4.3 Few-Shot Performance

Few-shot is a more challenging scenario with data scarcity, where only a few seen event types are available, and the model must handle a large number of unseen event types. In Table 2, we compare the performance of our model and the baselines on two datasets in zero-shot, 5-shot, and 10-shot settings. It can be observed that DAMPT performs well in different EAE learning scenarios. On ACE05-E, DAMPT achieves the maximal gains of 4.6%, 2.6%, and 2.1% in the zero-shot, 5-shot, and 10-shot settings, respectively. At the document level, DAMPT is also promising and attains gains across the board on the dataset WIKIEVENTS.

K-Shot Performance Intuitively speaking, models are supposed to be improved with more seen examples for unseen event types (e.g., 5-shot and 10-shot situations). It is worth noting that the improvement of our DAMPT is not as significant as that of other models, such as PAIE. From zero-shot to 10-shot, the results improve by 5.9%, while PAIE shows an enhancement of 11.2%. This is because our Slot-Transfer strategy can help DAMPT obtain well-informed module indicator tokens and slots in zero-shot scenarios, which are available for other models merely during few-shot learning. Another reason may be related to the expansion of seen examples, the obtained tunable components tend to be stable, resulting in less impact on performance improvement. These observations indicate that our model can gain benefits in few-shot settings with a small value of seen examples. Further analysis will be conducted in Section 4.6.

Influence of LLM It can be observed in Table 2 that in the zero-shot setting, DAMPT with the event templates generated by GPT-3.5 outperforms DAMPT with manually designed templates by 3.4% on ACE05-E and 0.9% on WIKIEVENTS. This may be attributed to the promising language ability of LLM in generating event descriptions.

4.4 Ablation Studies

We conduct ablation experiments on ACE05-E to demonstrate the impact of each component in our model. Each component is removed separately: **(1) w/o TypeSlot**: removing the event top-level Type Slots that transfer type knowledge; **(2) w/o RoleSlot**: removing the event Role Slots that transfer role knowledge; **(3) w/o TransEnc**: removing the Transformer Encoder that captures interaction among arguments; and **(4) w/o DMP**: without Dynamic Modular Prompt, that is, a single event template is provided to the model as a prompt as that in most models. Moreover, we also evaluate a variant of a base model with the addition of DMP (w/ DMP) for further evaluation.

As shown in Table 3, we can see that **(1)** each component of our proposed model plays its important and specific role in EAE learning; **(2)** the Transformer Encoder exhibits remarkable performance in the zero-shot setting, underscoring the significance of capturing interaction information among arguments, even without new type events; **(3)** TypeSlot and RoleSlot prove to be effective in low-resource environments, affirming the value of transferring event-specific knowledge from a minimal dataset; **(4)** Dynamic Modular Prompt serves

Model	Event Type Slot	Event Role Slot	Transformer Encoder	Dynamic Modular Prompt	Zero-Shot	Arg-C 1% Data	3% Data
DAMPT	✓	✓	✓	✓	62.2	27.4	39.1
-w/o TypeSlot	×	✓	✓	✓	61.1	20.9	37.5
-w/o RoleSlot	✓	×	✓	✓	60.1	24.0	36.3
-w/o TransEnc	✓	✓	×	✓	58.9	22.4	37.7
-w/o DMP	×	×	×	×	56.0	19.2	34.4
-w/ DMP	×	×	×	✓	59.3	24.9	36.9

Table 3: Ablation studies on ACE05-E in the data-scarce settings.

Contexts	DAMPT(ours)	PAIE
Sentence1 (*Personnel.Start-Position*): In Paris , the French media group said parent company chairman Jean - Rene Fourtou will <t> replace </t> Diller as chairman and chief executive of US unit.	*Person*:Jean - Rene Fourtou ✓	*Person*: Diller ×
Sentence2 (*Conflict.Demonstrate*):As I said , the officers did tell me that this is the largest pro - troops <t> demonstration </t> that has ever been in San Francisco since the Vietnam War.	*Place*:San Francisco ✓	*Place*: ∅ ×
Sentence3 (*Life.Marry*): Giuliani's first <t> marriage </t> to his second cousin, Regina Peruggi, lasted about 14 years.	*Person*: Giulian ✓ *Person*: cousin ✓	*Person*: Giulian ✓ *Person*: ∅ ×

Table 4: Examples of how DAMPT and PAIE perform.

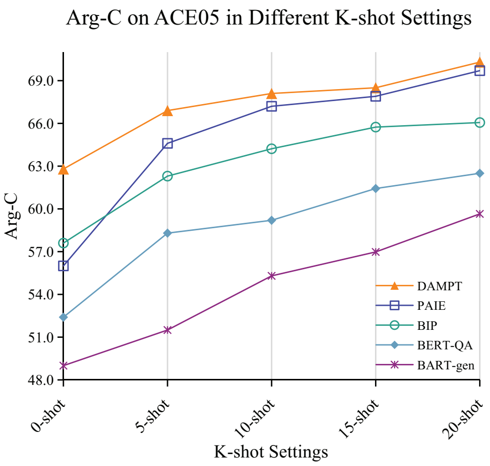

Figure 4: Analysis on different K-shot settings

as the foundation of our model. A certain improvement is achieved when using DMP alone in PAIE. This demonstrates that the additional event information provided by Event Type Module and the tunable module indicator tokens in Dynamic Modular Prompt work with EAE learning.

4.5 Case Studies

In order to showcase the EAE ability of our method, we sample several contexts from ACE05-E dataset to compare the extraction results of DAMPT and the baselines in Table 4. In **Sentence1**, due to the role-specific knowledge of *Person* transferred from seen events (e.g., *Personnel.Elect*), DAMPT can extract the argument **"Jean - Rene Fourtou"** when there is a distracting word **"Diller"**. In contrast, PAIE extracts the wrong argument. The benefit of equipping Slot-Transfer is also shown in **Sentence2**, where the knowledge transferred for *Place* improves the understanding ability of DAMPT. In **Sentence3**, as arguments **"Giuliani"** and **"cousin"** are associated with the word **"to"**, knowing that **"Giuliani"** serves as *Person* helps to easily extract **"cousin"** as *Person* when their interaction implied by preposition **"to"** is captured. With the argument interaction module, DAMPT can extract the second *Person* argument correctly compared to PAIE.

4.6 A Further Analysis on K-Shot Settings

We further conduct experiments on ACE05 where K increases from 0 to 20. We compare the Arg-C results between DAMPT and several baselines in Figure 4. DAMPT consistently dominates the other methods with increasing K. We observe that the performance gap between DAMPT and the other models is largest when K is equal to 0, and there is a shrinking trend when K increases, indicating that DAMPT is more suitable for the few-shot scenario with a small K. This also suggests our Dynamic Modular Prompt with Slot-Transfer algorithm can explore generalized and event-specific knowledge contained by a limited number of available events.

5 Conclusion

In this paper, we have proposed a novel fully automated prompt construction called DAMPT for data-scarce EAE. We have introduced Dynamic Modular Prompt which incorporates learnable indicator tokens to transfer generalized knowledge. We have also introduced Role Slot and Type Slot which enable transferring event-specific knowledge from a few event annotations. Moreover, we have incorporated Transformer encoder to capture argument interactions. Our evaluations in data-scarce settings demonstrate the effectiveness of DAMPT.

Limitations

Our proposed model explores the language understanding ability of pre-trained language models to tune the Dynamic Modular Prompt and generate spans selectors. As a result, the performance of our DAMPT is subject to the pre-training ability of pre-trained language models. It suggests a promising way in the future to generalize EAE capability during the pre-training phases.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

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*, pages 167–176.

Lu Dai, Bang Wang, Wei Xiang, and Yijun Mo. 2022. Bi-directional iterative prompt-tuning for event argument extraction. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6251–6263.

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)*. 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.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 3816–3830.

Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. 2021. WARP: Word-level Adversarial ReProgramming. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 4921–4933.

I-Hung Hsu, Kuan-Hao Huang, Elizabeth Boschee, Scott Miller, Prem Natarajan, Kai-Wei Chang, and Nanyun Peng. 2022. DEGREE: A data-efficient generation-based event extraction model. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1890–1908.

I-Hung Hsu, Zhiyu Xie, Kuan-Hao Huang, Prem Natarajan, and Nanyun Peng. 2023. AMPERE: AMR-aware prefix for generation-based event argument extraction model. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pages 10976–10993. 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*, pages 2160–2170.

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.

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.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 4582–4597.

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.

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.

Jian Liu, Yufeng Chen, and Jinan Xu. 2021. 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.

Jian Liu, Chen Liang, Jinan Xu, Haoyan Liu, and Zhe Zhao. 2023a. Document-level event argument extraction with a chain reasoning paradigm. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pages 9570–9583. Association for Computational Linguistics.

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

Xiao Liu, Heyan Huang, Ge Shi, and Bo Wang. 2022. Dynamic prefix-tuning for generative template-based event extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 5216–5228.

Qing Lyu, Hongming Zhang, Elior Sulem, and Dan Roth. 2021. Zero-shot event extraction via transfer learning: Challenges and insights. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 322–332.

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. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 6759–6774.

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.

Yubing Ren, Yanan Cao, Ping Guo, Fang Fang, Wei Ma, and Zheng Lin. 2023. Retrieve-and-sample: Document-level event argument extraction via hybrid retrieval augmentation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pages 293–306.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30.

Runxin Xu, Peiyi Wang, Tianyu Liu, Shuang Zeng, Baobao Chang, and Zhifang Sui. 2022. A two-stream AMR-enhanced model for document-level event argument extraction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5025–5036.

Bishan Yang and Tom M. Mitchell. 2016. Joint extraction of events and entities within a document context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 289–299.

Yunzhi Yao, Shengyu Mao, Ningyu Zhang, Xiang Chen, Shumin Deng, Xi Chen, and Huajun Chen. 2023. Schema-aware reference as prompt improves data-efficient knowledge graph construction. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 911–921.

Hongming Zhang, Haoyu Wang, and Dan Roth. 2021. Zero-shot Label-aware Event Trigger and Argument Classification. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1331–1340.

Kaihang Zhang, Kai Shuang, Xinyue Yang, Xuyang Yao, and Jinyu Guo. 2023. What is overlap knowledge in event argument extraction? APE: A cross-datasets transfer learning model for EAE. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pages 393–409. Association for Computational Linguistics.

Zhisong Zhang, Emma Strubell, and Eduard Hovy. 2022. Transfer learning from semantic role labeling to event argument extraction with template-based slot querying. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2627–2647.

Zexuan Zhong, Dan Friedman, and Danqi Chen. 2021. Factual probing is [MASK]: Learning vs. learning to recall. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5017–5033.

A Implementation Details

The architecture we use to construct DAMPT is BART-base[5], with 139 million parameters, consisting of 6 Transformer layers. All experiments are conducted on NVIDIA TITAN Xp GPU. We reported the average F1 score over five different random seeds to alleviate the negative impact of random training. Table 5 shows the statistics of datasets in data-scarce settings. Table 6 shows the detailed training configurations in DAMPT's training process.

B GPT-Generated Templates

B.1 Prompt for In-Context Learning

We list the in-context learning text for GPT-3.5 event templates generation for ACE05 in Figure 5. When performing in-context learning on the WIKIEVENTS dataset with numerous event

[5] https://huggingface.co/facebook/bart-base

Examples of Events:
event types: Conflict.Attack
event roles: Place, Target, Attacker, Instrument,
output: prompt start, Attacker (and Attacker) attacked Target (and Target) hurting victims using Instrument (and Instrument) at Place (and Place) ,end

event types: Movement.Transport
event roles: Vehicle, Artifact, Destination, Agent, Origin, Price,
output: prompt start, Agent (and Agent) transported Artifact (and Artifact) in Vehicle (and Vehicle) cost Price from Origin place (and Origin) to Destination place (and Destination , Destination) ,end
......

Describe all following Events(including Events showed in Examples) in concise terms (fill output):
event types: Conflict.Attack
event roles: Place, Target, Attacker, Instrument,
output:{output}
event types: Movement.Transport
event roles: Vehicle, Artifact, Destination, Agent, Origin, Price,
output:{output}
......

Figure 5: Context for GPT-3.5 in-context learning

Datasets	ACE05-E	WIKIEVENTS
#Event Types	33	50
#Event Roles	27	80
Full #Sent		
Train	17,172	5,262
Dev	923	378
Test	832	492
Low-resource(1%) #Sent		
Train	171	52
Dev	923	378
Test	832	492
Zero-shot #Sent		
Train	3,497	2,697
Dev	389	300
Test	1169	954

Table 5: Statistics of the datasets

Hyperparameter	Value
Optimizer	AdamW
Adam epsilon	1e-8
Learning rate	2e-5
Weight decay	0.01
Batch size	16(ACE05)/4(WIKIEVENTS)
Training steps	600(1% training data) 10000(30% training data/few-shot)
Max encoder length	192(ACE05)/500(WIKIEVENTS)
Max decoder length	80

Table 6: Hyperparameter settings

types, we split all types into two parts and generated templates separately in case of exceeding the context length limitation of ChatGPT. During the training and inference processes, we excluded in-context examples in the prompt and the length of prompts was limited within 512 tokens for the pre-trained language models.

B.2 Examples of Templates

We sample some templates generated by GPT and list them below:

Conflict.Attack: Attacker attacked Target using Instrument at Place.

Life.Die: Victim was killed by Agent using Instrument at Place.

Personnel.Start-Position: Person started working at Position of Entity organization at Place.

Business.Start-Org: Org was started by Agent at Place.

Contact.Meet: Entity met with Entity at Place.

Movement.Transport: Agent transported Artifact in Vehicle from Origin to Destination for Price.

Justice.Sentence: Defendant was sentenced for Crime by Adjudicator for Sentence at Place.

Transaction.Transfer-Money: Giver gave Money to Recipient for the benefit of Beneficiary at Place.

C EAE Performance of LLM

To demonstrate the performance of the LLM for a more comprehensive comparison, we have conducted experiments on a small set of ACE05-E test data (60 samples) with GPT-4 as a zero-shot solu-

model	P	R	F
GPT-4	37.8	53.8	44.5

Table 7: Performance of GPT-4 for EAE

tion. The results recorded in the Table 7 demonstrate that Few-shot learning and zero-shot learning remain challenging even for powerful models.

As demonstrated in Table 1 and Table 2, DAMPT shows significant F1-score improvements relative to the baselines in the low-resource and few-shot scenarios. The comparatively low F1 scores are attributed to the inherent data limitation in data-scarce scenarios.