AMPERE: AMR-Aware Prefix for Generation-Based Event Argument Extraction Model

I-Hung Hsu[*1] **Zhiyu Xie**[*3] **Kuan-Hao Huang**[2]
Premkumar Natarajan[1] **Nanyun Peng**[2]
[1]Information Science Institute, University of Southern California
[2]Computer Science Department, University of California, Los Angeles
[3]Computer Science Department, Tsinghua University
{ihunghsu, pnataraj}@isi.edu {khhuang, violetpeng}@cs.ucla.edu
xiezy19@mails.tsinghua.edu.cn

Abstract

Event argument extraction (EAE) identifies event arguments and their specific roles for a given event. Recent advancement in generation-based EAE models has shown great performance and generalizability over classification-based models. However, existing generation-based EAE models mostly focus on problem re-formulation and prompt design, without incorporating additional information that has been shown to be effective for classification-based models, such as the abstract meaning representation (AMR) of the input passages. Incorporating such information into generation-based models is challenging due to the heterogeneous nature of the natural language form prevalently used in generation-based models and the structured form of AMRs. In this work, we study strategies to incorporate AMR into generation-based EAE models. We propose AMPERE, which generates AMR-aware prefixes for every layer of the generation model. Thus, the prefix introduces AMR information to the generation-based EAE model and then improves the generation. We also introduce an adjusted copy mechanism to AMPERE to help overcome potential noises brought by the AMR graph. Comprehensive experiments and analyses on ACE2005 and ERE datasets show that AMPERE can get $4\% - 10\%$ absolute F1 score improvements with reduced training data and it is in general powerful across different training sizes.

1 Introduction

Event argument extraction (EAE) aims to recognize event arguments and their roles in an event. For example, in Figure 1, EAE models need to extract *districts*, *u.s. supreme court*, and *washington* and the corresponding roles — *Plaintiff*, *Adjudicator*, and *Place* for the *Justice:Appeal* event with trigger *appeal*. EAE has long been a challenging task in NLP, especially when training data is limited (Wang et al., 2019; Ma et al., 2022). It is an important task for various downstream applications (Zhang et al., 2020; Berant et al., 2014; Hogenboom et al., 2016; Wen et al., 2021; Wu et al., 2022).

Recently, generation-based EAE models (Hsu et al., 2022a; Lu et al., 2021; Li et al., 2021; Paolini et al., 2021; Parekh et al., 2022) are proposed and have shown great generalizability and competitive performance compared to traditional classification-based methods (Chen et al., 2015; Ma et al., 2020; Hsu et al., 2022b; Fincke et al., 2022). However, existing generation-based EAE models mostly focus on problem reformulation and prompt design without incorporating auxiliary syntactic and semantic information that is shown to be effective in classification-based methods (Huang et al., 2016; Xu and Huang, 2022; Huang et al., 2018; Ahmad et al., 2021; Veyseh et al., 2020).

In this work, we explore how to incorporate auxiliary structured information into generation-based EAE models. We focus on abstract meaning representation (AMR) (Banarescu et al., 2013), which extracts rich semantic information from the input sentence. As the Figure 1's example shows, AMR graph summarizes the semantic structure of the input passage, and many of its nodes and edges share strong similarities with the event structures. For example, the trigger word *appeal* can be mapped to the node *"appeal-01"*, and the subject who appeals can be found using edge *"ARG0"*. Hence, the AMR graph could provide important clues for models to figure out event arguments, resulting in performance improvements (Zhang and Ji, 2021) and better generalizability (Huang et al., 2018) for classification-based methods. However, it is unclear how to best integrate AMR into generation-based methods. The heterogeneous nature between the AMR graph and the natural language prompts[1] in the generation-based EAE models causes the

[*]The authors contribute equally.

[1]For example, event type description and target generation templates.

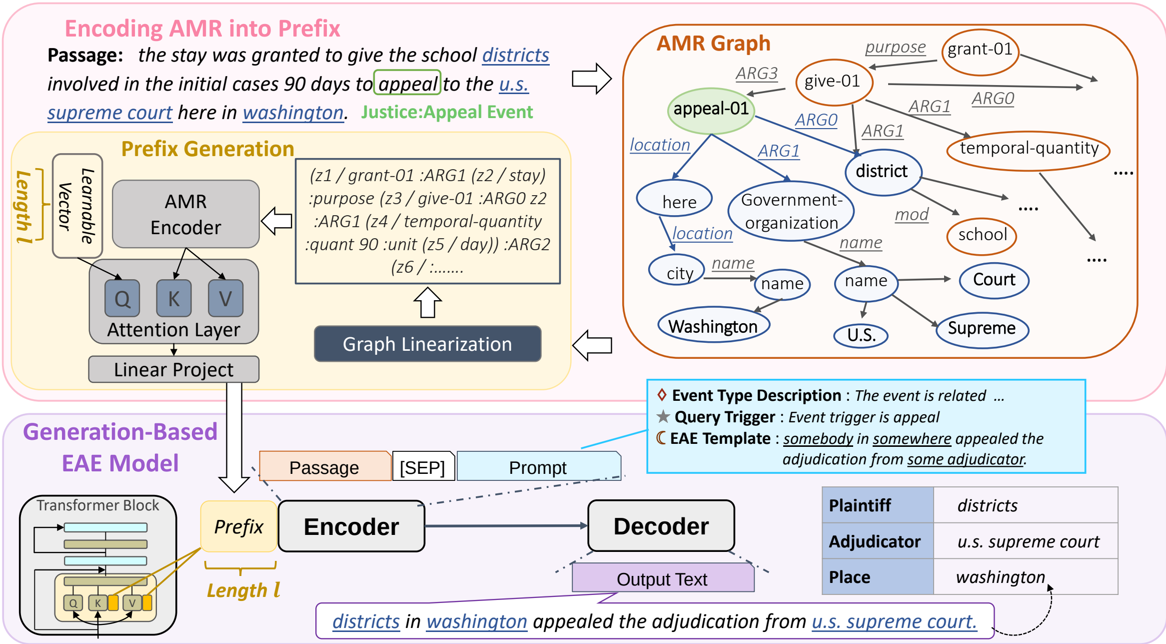

Figure 1: An overview of AMPERE using an example from the ACE 2005 dataset. Given a passage and an event trigger, we first use an AMR parser to obtain the AMR graph of the input passage. The linearized AMR graph sequence will be encoded into a l-length prefix by an AMR encoder and an attention layer. Our generation-based EAE model equipped with the AMR-aware prefix then summarizes the event mentioned in the passage into a natural sentence that follows a pre-defined template in the prompt. The final arguments and the corresponding roles can be extracted from the generated sentence.

difficulty of the model design.

To overcome the challenge, we propose AMPERE (**AM**r-aware **P**refix for generation-based **E**vent a**R**gument **E**xtraction), which encodes AMR graph into prefix (Li and Liang, 2021) to regulate the generation-based EAE models. Specifically, an additional AMR encoder is used to encode the input AMR graph into dense vectors. Then, these vectors will be disassembled and distributed to every Transformer layer in generation-based EAE models as the prefix. These generated prefixes are transformed into additional key and value matrices to influence the attention calculation, hence, guiding the generation.

We also introduce an adjusted copy mechanism for AMPERE to overcome potential noises brought by the AMR graph. Specifically, as we can observe in Figure 1, AMR parsers will include additional normalization (turning *washington* into *Washington*) and word disambiguation (using *appeal-01* rather than *appeal*) to create AMR graphs. Such normalization could impact the generation to produce some words that are not in the original input, especially when the training data is limited. Hence, we apply a copy mechanism (See et al., 2017) and add an additional regularization loss term to encourage copying from the input passage.

We conduct experiments on ACE 2005 (Doddington et al., 2004) and ERE (Song et al., 2015) datasets using different ratios of training data. Our results show that AMPERE outperforms several prior EAE works in both datasets. Under low-resource settings that use only 5% or 10% of training data, we can get $4\% - 10\%$ absolute F1-scores of improvement, and our method is in general powerful across different training sizes and different datasets. We also present a comprehensive study of different ways to incorporate AMR information into a generation-based EAE model. We will show that AMPERE is the best way among the various methods we explored. Our code can be found at `https://github.com/PlusLabNLP/AMPERE`.

2 Method

AMPERE uses DEGREE (Hsu et al., 2022a) as the base generation-based EAE model [2] (Section 2.1), and augments it with AMR-aware prefixes, as shown in Figure 1. To generate the AMR-aware prefixes, we first use a pre-trained AMR parser to obtain the AMR graph of the input sentence (Sec-

[2]We use the EAE version of DEGREE (Hsu et al., 2022a).

tion 2.2). Then, the graph is transformed into dense vectors through graph linearization and an AMR encoder. Then, these dense vectors will be disassembled and distributed to each layer of our base generation-based EAE model so the generation is guided by the AMR information (Section 2.3). Finally, we introduce the training loss for AMPERE and our adjusted copy mechanism that can help AMPERE overcome additional noise brought from AMR graphs (Section 2.4).

2.1 Generation-Based EAE Model

Despite our AMR-aware prefix being agnostic to the used generation-based EAE model, we select DEGREE (Hsu et al., 2022a) as our base model because of its great generalizability and performance. Here, we provide a brief overview of the model.

Given a passage and an event trigger, DEGREE first prepares the *prompt*, which includes an event type description (a sentence describing the trigger word), and an event-type-specific template, as shown in Figure 1. Then, given the passage and the prompt, DEGREE summarizes the event in the passage following the format of the EAE template, so that final predictions can be decoded easily by comparing the template and the output text. Take the case in Figure 1 as an example, by comparing *"districts in washington appealed the adjudication from u.s. supreme court."* with the template *"somebody in somewhere appealed the adjudication from some adjudicator."*, we can know that the "districts" is the argument of role *"Plaintiff"*. This is because the corresponding placeholder *"somebody"* of the role *"Plaintiff"* has been replaced by *"districts"* in the model's prediction.

2.2 AMR Parsing

The first step of our method is to prepare the AMR graph of the input passage. We consider SPRING (Bevilacqua et al., 2021), a BART-based AMR parser trained on AMR 3.0 annotation, [3] to be our AMR parser. As illustrated by Figure 1, the AMR parser encodes the input sentence into an AMR graph, which is a directed graph where each node represents a semantic concept (e.g., *"give-01"*, *"appeal-01"*) and each edge describe the categorical semantic relationship between two concepts (e.g., *ARG0*, *location*) (Banarescu et al., 2013).

[3] `https://catalog.ldc.upenn.edu/LDC2020T02`

2.3 AMR-Aware Prefix Generation

Our next step is to embed the information into prefixes (Li and Liang, 2021) for our generation-based EAE model. To encode the AMR graph, we follow Konstas et al. (2017) to adopt a depth-first-search algorithm to linearize the AMR graph into a sequence, as shown in the example in Figure 1. Then, an AMR encoder is adapted to encode the representation of the sequence. One of the advantages of our method is the flexibility to use models with different characteristics to our generation-based EAE model to encode AMR. Here, we consider two AMR encoders to form different versions of AMPERE:

- AMPERE (`AMRBART`): We consider using the encoder part of the current state-of-the-art AMR-to-text model — `AMRBART` (Bai et al., 2022) that pre-trained on AMR 3.0 data. [4] The model is based on `BART-large` and its vocabulary is enlarged by adding all relations and semantic concepts in AMR as additional tokens. Employing the model as our AMR encoder enables AMPERE to leverage knowledge from other tasks.
- AMPERE (`RoBERTa`): `RoBERTa-large` (Liu et al., 2019b) is also considered as our AMR encoder as pre-trained masked language models are typical choices to perform encoding tasks. In order to make `RoBERTa` better interpret the AMR sequence, we follow Bai et al. (2022) to add all relations in AMR (e.g. *ARG0*, *ARG1*) as special tokens. However, since the model is not pre-trained on abundant AMR-to-text data, we do not include semantic concepts (e.g. concepts end with *-01*) as extra tokens. [5]

After getting the representation of the linearized sequence, we then prepare l learnable vectors as queries and an attention layer, where l is a hyper-parameter that controls the length of the used prefixes. These queries will compute attention with the representations of the linearized AMR sequence, then, we will obtain a set of compressed dense vector $\mathbf{P}$. This $\mathbf{P}$ will be transformed into the prefixes (Li and Liang, 2021) that we will inject into our generation-based EAE model.

To be more specific, we first disassemble $\mathbf{P}$ into L pieces, where L is the number of lay-

[4] `https://github.com/goodbai-nlp/AMRBART`

[5] If adding semantic concepts as extra tokens, `RoBERTa` will loss the ability to grasp its partial semantic meaning from its surface form, such as understanding that *"appeal-01"* is related to *"appeal"*.

ers in the base generation-based EAE model, i.e., $\mathbf{P} = \{P^1, P^2, ...P^L\}$. Then, in the n-th layer of the EAE model, the prefix is separated into two matrices, standing for the addition key and value matrices: $P^n = \{K^n, V^n\}$, where K^n & V^n are the addition key and value matrices, and they can be further written as $K^n = \{\mathbf{k}_1^n, ..., \mathbf{k}_l^n\}$ and $V^n = \{\mathbf{v}_1^n, ..., \mathbf{v}_l^n\}$. $\mathbf{k}_*$ and $\mathbf{v}_*$ are vectors with the same hidden dimension in the Transformer layer. These additional key and value matrices will be concatenated with the original key and value matrices in the attention block. Therefore, when calculating dot-product attention, the query at each position will be influenced by these AMR-aware prefixes. The reason of generating layer-wise queries and keys is to exert stronger control. We generate layer-wise key-value pairs as each layer may embed different information. These keys influence the model's weighting of representations towards corresponding generated values. Empirical studies on layer-wise versus single-layer control can be found in Liu et al. (2022b).

It is worth noting that Li and Liang (2021)'s prefix tuning technique uses a fixed set of prefixes disregarding the change of input sentence, AMPERE will *generate* a different set of prefixes when the input passage varies. And the variation reflects the different AMR graph's presentation.

We can inject prefixes into the encoder self-attention blocks, decoder cross-attention blocks, or decoder self-attention blocks in our generation-based EAE model. Based on our preliminary experiments, we observe that using prefix in encoder self-attention blocks and decoder cross-attention blocks works best in AMPERE.

2.4 Adjusted Copy Mechanism

We follow DEGREE's setting to use `BART-large` (Lewis et al., 2020) as the pre-trained generative model, and the training objective of our generation-based EAE model is to maximize the conditional probability of generating a ground-truth token given the previously generated ones and the input context in the encoder $x_1, x_2, ..x_m$:

$$Loss = -\log(\sum_i P(y_i|y_{<i}, x_1, ..., x_m)), \tag{1}$$

where y_i is the output of the decoder at step i. In DEGREE's setting, the probability of predicting an token t fully relies on the generative model. Although this setting is more similar to how `BART-large` is pre-trained and thus better leverages the power of pre-training, the loose constraints on the final prediction could generate hallucinated texts (Ji et al., 2022) or outputs not following the template. Such an issue could be enlarged if less training data is used and more input noise is presented, such as when incorporating AMR graphs.

To enhance the control, one commonly-used technique is to apply copy mechanism (See et al., 2017) to generation-based event models (Huang et al., 2022, 2021). , i.e.,

$$\begin{aligned} &P(y_i = t|y_{<i}, x_1, .., x_m) = \\ &\quad w_{gen}^i P_{gen}(y_i = t|y_{<i}, x_1, ..., x_m))+ \\ &\quad (1 - w_{gen}^i)(\sum_{j=0}^{m} P_{copy}^i(j|y_{<i}, x_1, ..., x_m) \times \mathbb{1}(x_j = t)), \end{aligned} \tag{2}$$

where $w_{gen}^i \in [0, 1]$ is the probability to generate, computed by passing the last decoder hidden state to an additional network. $P_{copy}^i(j|\cdot)$ is the probability to copy input token x_j, and it's computed by using the cross-attention weights in the last decoder layer at time step i. When $w_{gen}^i = 1$, it is the original model used by DEGREE, while if $w_{gen}^i = 0$, this model will only generate tokens from the input.

Our core idea of the adjusted copy mechanism is to encourage the model to copy more, and this is achieved by introducing a regularization term on w_{gen}^i to the loss function of AMPERE:

$$\begin{aligned} &Loss_{\text{AMPERE}} = \\ &\quad -\log(\sum_i P(y_i|y_{<i}, x_1, ..., x_m)) + \lambda \sum_i w_{gen}^i, \end{aligned} \tag{3}$$

where λ is a hyper-parameter. Compared to fully relying on copy from input, our method still allows the generative model to freely generate tokens not presented in the input. Compared to ordinary copy mechanisms, the additional regularizer will guide the model to copy more. Using this loss, we train the whole AMPERE end-to-end.

3 Experiments

We conduct experiments to verify the effectiveness of AMPERE. All the reported numbers are the average of the results from three random runs.

3.1 Experimental Settings

Datasets and Data split. We adopt the event annotation in ACE 2005 dataset (Doddington et al., 2004) (**ACE05-E**)[6], and the English split in ERE

[6] `https://catalog.ldc.upenn.edu/LDC2006T06`

Model	Type	Development Set						Test Set					
		5%	10%	20%	30%	50%	100%	5%	10%	20%	30%	50%	100%
Argument Classification F1-Score (%) in ACE05-E													
DyGIE++ (Wadden et al., 2019)	Cls	34.6	48.5	52.5	57.5	57.9	60.0	29.3	42.4	49.5	53.2	54.5	57.4
OneIE (Lin et al., 2020)	Cls	38.6	56.0	63.2	67.6	70.4	71.8	34.6	50.0	59.6	63.0	68.4	70.6
Query and Extract (Wang et al., 2022)	Cls	10.5	27.7	37.6	50.0	54.6	61.7	11.0	20.9	34.3	44.3	49.6	59.1
AMR-IE (Zhang and Ji, 2021)	Cls	40.0	56.3	61.3	67.4	70.6	73.1	36.8	48.5	58.3	62.6	66.1	70.3
PAIE (Ma et al., 2022)	Gen	46.6	57.6	64.6	69.3	70.3	74.1	46.3	56.3	62.8	65.8	69.1	72.1
DEGREE (Hsu et al., 2022a)	Gen	41.4	56.8	62.5	68.9	70.5	73.8	41.7	57.7	58.9	65.8	68.2	73.0
AMPERE (AMRBART)	Gen	52.3	**61.5**	**67.2**	71.2	**72.7**	75.5	52.4	61.0	**66.4**	**69.7**	71.1	73.4
AMPERE (RoBERTa)	Gen	**53.2**	**61.5**	66.6	**71.8**	72.5	**76.6**	**53.4**	**61.7**	**66.4**	69.5	**71.9**	**74.2**
Argument Classification F1-Score (%) in ERE-EN													
DyGIE++ (Wadden et al., 2019)	Cls	42.2	45.4	49.0	50.1	51.5	56.8	40.0	44.6	49.5	52.0	53.7	56.0
OneIE (Lin et al., 2020)	Cls	51.4	59.5	62.0	65.6	68.6	71.2	49.5	56.1	62.3	66.1	67.7	70.1
Query and Extract (Wang et al., 2022)	Cls	22.0	37.3	41.2	49.4	57.0	65.0	19.7	34.0	42.4	50.1	57.7	64.3
AMR-IE (Zhang and Ji, 2021)	Cls	44.8	55.2	56.8	65.2	67.6	70.1	44.1	53.7	60.4	65.6	68.9	71.5
DEGREE (Hsu et al., 2022a)	Gen	57.2	62.5	63.9	67.1	70.2	73.3	57.5	63.9	67.4	69.1	**73.3**	74.9
AMPERE (AMRBART)	Gen	62.4	**66.8**	**66.6**	68.8	**70.8**	73.6	62.9	66.7	**68.5**	**71.3**	72.5	**75.4**
AMPERE (RoBERTa)	Gen	**63.1**	66.7	**66.6**	**69.7**	70.6	**73.8**	**63.2**	**67.7**	68.4	70.5	72.5	75.0

Table 1: Argument classification F1-scores (%) under different data proportion settings for ACE05-E and ERE-EN datasets. The highest scores are in bold and the second-best scores are underlined. Generation-based models and Classification-based models are indicated by "Gen" and "Cls" respectively. Due to space constraints, the table with argument identification F1-scores is listed in Appendix §C.

dataset (Song et al., 2015) (**ERE-EN**)[7]. ACE 2005 contains files in English, Chinese, and Arabic, and ERE includes files in English and Chinese. In this paper, we only use the documents in English, and split them to sentences for use in our experiments. We follow prior works (Wadden et al., 2019; Lin et al., 2020) to preprocess each dataset. After preprocessing, **ACE05-E** has 33 event types and 22 argument roles, and **ERE-EN** are with 38 event types and 21 argument roles in total. Further, we follow Hsu et al. (2022a) to select 5%, 10%, 20%, 30%, and 50% of training samples to generate the different data split as the training set for experiments. The data statistics are listed in Table 4 in the Appendix A.[8]

Evaluation metrics. We report the F1-score for argument predictions following prior works (Wadden et al., 2019; Lin et al., 2020). An argument is correctly identified (**Arg-I**) if the predicted span matches the span of any gold argument; it is correctly classified (**Arg-C**) if the predicted role type also matches.

Implementation details. We use the AMR tools as we mentioned in Section 2. When training our models, we set the learning rate to 10^{-5}. The number of training epochs is 60 when training on ACE05E, and 75 when training on ERE-EN. We simply set λ as 1 for all our models. We do hyperparameter searching using the setting that trains on 20% of data in ACE05E and selects the best model based on the development set results. We set $l = 40$, and batch size is set to 4 for AMPERE (AMRBART) and 6 for AMPERE (RoBERTa) in the end. This is searching from $l = \{30, 40, 50\}$ and batch size $= \{4, 6, 8, 12\}$.

Baselines. We compare AMPERE with the following classification-based models: (1) **DyGIE++** (Wadden et al., 2019), which extracts information by scoring spans with contextualized representations. (2) **OneIE** (Lin et al., 2020), a joint IE framework that incorporates global features. (3) **Query and Extract** (Wang et al., 2022), which uses attention mechanisms to evaluate the correlation between role names and candidate entities. (4) **AMR-IE** (Zhang and Ji, 2021), which captures non-local connections between entities by aggregating neighborhood information on AMR

[7] https://catalog.ldc.upenn.edu/LDC2020T19

[8] The license for ACE 2005 and ERE is *LDC User Agreement for Non-Members*. In both datasets, event types like *Conflict:Attack* may include offensive content such as information related to war or violence. Some passages are extracted from broadcast news, thus some real names may appear in the data. Considering the datasets are not publicly available, and these contents are likely to be the arguments to extract in our task, we do not change the data for protecting or anonymizing.

graph, and designed hierarchical decoding based on AMR graph information. We also consider the following generation-based models: (5) **PAIE** (Ma et al., 2022), a framework that integrated prompt tuning, and generates span selectors for each role. [9] (6) **DEGREE** (Hsu et al., 2022a). The generation-based EAE model we used as our base model.

To ensure a fair comparison across models, we adopt the official codes of the above baselines to train them on the identical data and did hyper-parameter tuning. For all the classification-based methods, we use `RoBERTa-large`, and for all the generation-based methods, we use `BART-large` as the pre-trained language models. Appendix §B shows details about the implementation.

3.2 Results

Table 1 shows the argument classification (**Arg-C**) F1-scores in ACE05-E and ERE datasets under different data proportions. Overall, both AMPERE (`RoBERTa`) and AMPERE (`AMRBART`) consistently outperform all other baselines except the test set results of using 50% data in ERE-EN.

From the table, we can notice that AMPERE significantly outperforms our base model DEGREE in all experiments in ACE05-E, and in ERE-EN, the improvement is also considerable. When trained with less than 20% data in ACE05-E, AMPERE (`RoBERTa`) can consistently achieve more than 4 points of improvement over DEGREE in both the development and test sets. In the following Section 4, we will further discuss the detailed contribution of our method over DEGREE.

To quantitatively evaluate the effectiveness of AMR's incorporation, we can first check the performance of AMR-IE. AMR-IE achieves competitive performance among classification-based models, especially under extremely low-resource settings. This is coincident with how AMPERE's result shows. AMPERE outperforms both DEGREE and PAIE, and the gap is more obvious under low-resource settings. For example, in the 5% data proportion setting, AMPERE (`RoBERTa`) made over 11 points of improvement over DEGREE in ACE05-E's test Set. In the meanwhile, AMPERE (`RoBERTa`) achieves 4.4 points of performance gain compared with PAIE. All this shows the empirical evidence that AMR information can hint to the models' semantic structure of the input passage, and this is especially helpful for models when training samples are limited. Despite the strong performance of AMR-IE, AMPERE can still outperform it across all the settings, indicating the effectiveness of our method.

Comparing AMPERE (`AMRBART`) and AMPERE (`RoBERTa`), we show that our proposed method does not necessarily rely on pre-trained AMR-to-Text models. Particularly, AMPERE (`RoBERTa`), which employs a pre-trained `RoBERTa-large` as the AMR Encoder still achieves competitive results to AMPERE (`AMRBART`), which uses AMR-to-Text data. Yet, the advantage of using AMR-to-Text data as an auxiliary is that we can get similar results with less parameters. The *AMR encoder component* of AMPERE (`RoBERTa`) has approximately 1.7 times more parameters than that of AMPERE (`AMRBART`), as we only use the encoder part of `AMRBART` in AMPERE(AMRBART). Nevertheless, the pre-trained knowledge from AMR-to-text data enables AMPERE (`AMRBART`) to perform competitively with AMPERE (`RoBERTa`).

4 Analysis

In this section, we present comprehensive ablation studies and case studies to validate our model designs. Two essential parts of our design, the AMR-aware prefix, and the adjusted copy mechanism will be examined in the following studies. For all the experiments in this section, we use the setting of training on 5% and 20% ACE05-E data to simulate very low-resource and low-resource settings.

4.1 Different Ways for AMR Incorporation

We compare different ways to incorporate AMR information into generation-based EAE models:

- **AMR Prompts.** We follow the same process as AMPERE to obtain the linearized AMR graph sequence. Then, we directly concatenate the linearized AMR graph sequence to the input text as part of the prompts.
- **`AMRBART` Encoding Concatenation.** After obtaining the AMR sequence representations after the AMR encoder using `AMRBART`, we concatenate this encoding with the output representation in our generation-based EAE model and feed them together to the decoder.
- **`RoBERTa` Encoding Concatenation.** The method is similar to the AMRBART Encoding Concatenation method, except that we use

[9] PAIE requires manually designed prompts for model training, and ERE-EN dataset is not considered in their official codebase. Hence, we do not include PAIE as a baseline on ERE-EN.

Model	5% ACE05-E Data				20% ACE05-E Data			
	Dev. Set		Test Set		Dev. Set		Test Set	
	Arg-I	Arg-C	Arg-I	Arg-C	Arg-I	Arg-C	Arg-I	Arg-C
AMPERE w/o AMR-aware Prefix	$57.8_{\pm2.32}$	$49.8_{\pm2.59}$	$55.9_{\pm0.75}$	$47.5_{\pm0.32}$	$70.6_{\pm0.94}$	$64.5_{\pm0.30}$	$67.0_{\pm0.81}$	$62.6_{\pm1.36}$
AMPERE (AMRBART)	$59.9_{\pm1.99}$	$52.3_{\pm1.54}$	$59.8_{\pm2.00}$	$52.4_{\pm1.53}$	$72.0_{\pm0.80}$	$\mathbf{67.2}_{\pm0.55}$	$70.2_{\pm0.84}$	$\mathbf{66.4}_{\pm1.04}$
AMPERE (RoBERTa)	$62.1_{\pm1.73}$	$\mathbf{53.2}_{\pm2.26}$	$\mathbf{61.0}_{\pm0.98}$	$\mathbf{53.4}_{\pm0.21}$	$71.5_{\pm1.00}$	$66.6_{\pm0.12}$	$\mathbf{70.5}_{\pm1.28}$	$\mathbf{66.4}_{\pm0.86}$
AMPERE (AMRBART) w/ frozen AMR Encoder	$60.9_{\pm2.10}$	$51.5_{\pm1.78}$	$58.3_{\pm1.63}$	$51.1_{\pm1.21}$	$\mathbf{72.5}_{\pm0.50}$	$66.5_{\pm1.06}$	$70.0_{\pm0.37}$	$65.8_{\pm0.19}$
AMPERE (RoBERTa) w/ frozen AMR Encoder	$\mathbf{62.5}_{\pm1.49}$	$50.9_{\pm1.34}$	$60.6_{\pm0.46}$	$50.7_{\pm0.09}$	$71.7_{\pm0.50}$	$66.0_{\pm0.76}$	$69.8_{\pm1.52}$	$65.5_{\pm1.47}$
AMR Prompts	$56.7_{\pm1.00}$	$48.4_{\pm1.11}$	$55.2_{\pm1.33}$	$47.2_{\pm1.25}$	$71.2_{\pm0.66}$	$65.7_{\pm0.80}$	$69.5_{\pm0.26}$	$64.9_{\pm0.51}$
AMRBART Encoding Concatenation	$58.4_{\pm0.45}$	$50.3_{\pm1.01}$	$56.4_{\pm2.16}$	$48.3_{\pm1.50}$	$71.2_{\pm0.87}$	$64.7_{\pm0.14}$	$69.1_{\pm1.54}$	$64.4_{\pm1.33}$
RoBERTa Encoding Concatenation	$6.4_{\pm0.85}$	$4.8_{\pm1.10}$	$4.9_{\pm2.81}$	$3.3_{\pm1.96}$	$11.6_{\pm2.58}$	$8.7_{\pm0.55}$	$11.4_{\pm1.38}$	$10.4_{\pm1.69}$

Table 2: Ablation study of different ways for AMR incorporation. Report numbers in F1-scores (%).

RoBERTa as the AMR encoder. [10]

For comparison, we provide AMPERE's performance without any AMR incorporation as a baseline. Additionally, we also consider AMPERE with frozen AMR encoder[11] in the comparisons to exclude the concern of extra learnable parameters of AMPERE compared to baselines such as AMR Prompts. Note that all the mentioned models above are implemented with our proposed adjusted copy mechanism. Table 2 shows the results.

From the table, we observe that AMPERE gets the best performance among all the ways we explored and achieves 4.9% and 4.2% F1-score improvements over the model without AMR incorporation under the case of using 5% & 20% of training data, respectively.

An interesting finding is that the performance of AMR Prompts is worse than the model without any AMR incorporation in the very low-resource setting (5% data). As mentioned in Section 1, the heterogeneous nature between AMR graph and natural language sentences is an important intuitive for our model design. AMR often uses special tokens such as *:ARG0* or *appeal-01*, and in implementation like AMR Prompts, it would be confusing for models when training samples are not sufficient.

Furthermore, due to the heterogeneous vector space between AMRBART and RoBERTa, RoBERTa Encoding Concatenation method could not work well. In comparison, the prefix design of AMPERE shows strong adaptability, as AMPERE (AMRBART) and AMPERE (RoBERTa) both outperform the other implementation methods.

[10]We also explore the variation that we add a linear layer to the AMR encoding to help space alignment, but there is little performance difference on both AMRBART Encoding Concatenation & RoBERTa Encoding Concatenation.

[11]For these type of models, during training, the AMR encoder's parameter is fixed. Hence the number of learnable parameters is comparable to DEGREE.

Finally, we focus on the results from AMPERE with frozen AMR Encoder. We can observe that despite slight performance degradation compared to fully-trainable AMPERE, AMPERE with frozen AMR Encoder still obtain at least 1% absolute F1-scores improvements over other AMR incorporation methods.

4.2 Studies of Adjusted Copy Mechanism

To justify the effectiveness of our adjusted copy mechanism, we compare our adjusted copy mechanism with the following method:

- **AMPERE w/o any copy.** For comparison, we adopt a normal generation-based model adapted with AMR-aware prefixes.
- **AMPERE w/ pure copy.**: In Equation 2, we directly set $w^i_{gen} = 0$. In other words, tokens not presented in the input can not be generated.
- **AMPERE w/ ordinary copy mechanism.** We apply the copy mechanism but train the model with the loss function in Equation 1.

In Table 3, the experiment with AMPERE (AMRBART) and AMPERE (RoBERTa) lead to similar conclusions. Any kind of copy mechanism can lead to noticeable improvement, and the performance gap between methods with and without copy mechanism is larger in the lower data proportion setting. Our adjusted copy mechanism stably outperforms the other methods in studies. Compared to the traditional copy mechanism, our method encourages the model to copy more, hence can stably overcome the very low-resource challenges. Compared to fully relying on copy from input, our method allows the generative model to freely generate tokens not presented in the input, so as to better leverage the pre-trained language model's power, leading to better performance when data is slightly more available.

Model		5% ACE05-E Data				20% ACE05-E Data			
		Dev. Set		Test Set		Dev. Set		Test Set	
		Arg-I	Arg-C	Arg-I	Arg-C	Arg-I	Arg-C	Arg-I	Arg-C
AMPERE (AMRBART)	w/ adjusted copy mechanism	**59.9**±1.99	**52.3**±1.54	**59.8**±2.00	**52.4**±1.53	**72.0**±0.80	**67.2**±0.55	70.2±0.84	**66.4**±1.04
	w/o any copy	48.7±0.67	41.3±1.56	49.0±1.68	43.2±0.77	69.8±0.64	63.7±0.27	65.5±0.90	61.4±0.93
	w/ pure copy	57.0±1.25	50.4±2.76	55.0±1.59	49.0±0.84	70.0±0.89	65.6±0.80	67.3±1.77	62.8±1.79
	w/ ordinary copy mechanism	58.9±1.28	49.9±0.32	53.4±3.34	48.1±2.61	70.7±0.58	64.1±0.26	68.3±1.07	63.6±0.30
AMPERE (RoBERTa)	w/ adjusted copy mechanism	**62.1**±1.73	**53.2**±2.26	**61.0**±0.98	**53.4**±0.21	71.5±1.00	**66.6**±0.12	**70.5**±1.28	**66.4**±0.86
	w/o any copy	52.5±0.85	44.0±1.22	50.7±1.79	44.5±1.81	69.9±0.16	62.7±0.26	65.5±1.01	61.1±1.34
	w/ pure copy	56.9±1.63	48.5±1.08	55.5±1.62	48.6±0.50	71.1±0.67	66.3±1.20	67.3±1.32	63.7±1.28
	w/ ordinary copy mechanism	58.5±2.03	50.7±0.18	54.5±1.01	47.8±1.09	71.1±1.40	64.9±0.74	69.0±1.61	64.3±1.16

Table 3: The study of using different generation mechanisms. Report numbers in F1-scores (%). The best performance among methods using the same model architecture is highlighted in bold.

4.3 Case Study

4.3.1 Output Examples

To intuitively explain the benefit of our method over previous generation-based EAE models, we present examples here to showcase the influence of incorporating AMR information. We compare AMPERE and DEGREE that both trained on 20% ACE05-E data and demonstrate two examples in Figure 2 to show the difference of their generated output text.

Example A presents a case where the edges in the AMR graph helps the model to classify the correct role type of argument *"government"*. Without AMR information, DEGREE incorrectly predicts the *"government"* to be the agent that launched some organization. In the AMR graph, edge *ARG1* points to the object of the action concept *form-01*. Thus, in the AMPERE's output, *"government"* is correctly classified as the object of *"form"*.

Example B in Figure 2 shows how the AMR graph hints AMPERE about the argument *"judge"*. By looking up the subject of verb *"order"* in the AMR graph, the model is able to find the adjudicator of the event. Thus, AMPERE could correctly replace the adjudicator placeholder in the template with real adjudicator, *"judge"*.

4.3.2 Error Analysis

To point out future research direction for generation-based EAE models, we performed error analysis on 30 cases where our AMPERE (RoBERTa) made mistakes. We identified two common types of errors: (1) ambiguous span boundaries, and (2) incorrect distinction between events of the same type.

For instance, in the case of "ambiguous span boundaries," AMPERE (RoBERTa) incorrectly predicted *"Christian Ayub Masih"* instead of the correct label *"Ayub Masih."* We observe that generation-based models struggle to accurately predict span boundaries, as both AMPERE (RoBERTa)'s output and the ground truth can fit into the sentence template coherently. Even with the inclusion of AMR, the model's ability to identify potential boundaries from the AMR graph through learning remains limited.

Regarding the issue of "incorrect distinction between events of the same type," we present an example to illustrate this. In the given input sentence, *"As well as previously holding senior positions at Barclays Bank, BZW and Kleinwort Benson, McCarthy was formerly a top civil servant at the Department of Trade and Industry."*, the model becomes confused between the two "Personnel:End-Position" events, each triggered by *"previousl"* and *"formerly"*, respectively, due to subtle differences. We suggest that incorporating additional structural knowledge, such as dependency parsing information, to separate the sentences structurally, could be a potential solution. However, we leave this research as future works.

5 Related Work

Generation-based event (argument) extraction models. Traditionally, most models for EAE are classification-based (Chen et al., 2015; Ma et al., 2020; Hsu et al., 2022b; Fincke et al., 2022). Recently, generation-based EAE models (Hsu et al., 2022a; Lu et al., 2021; Li et al., 2021; Paolini et al., 2021) become more and more popular due to their flexibility to present different output structures (Yan et al., 2021), to be unified considered with similar tasks (Lu et al., 2022), and their competitive performance (Hsu et al., 2022a; Liu et al., 2022a).

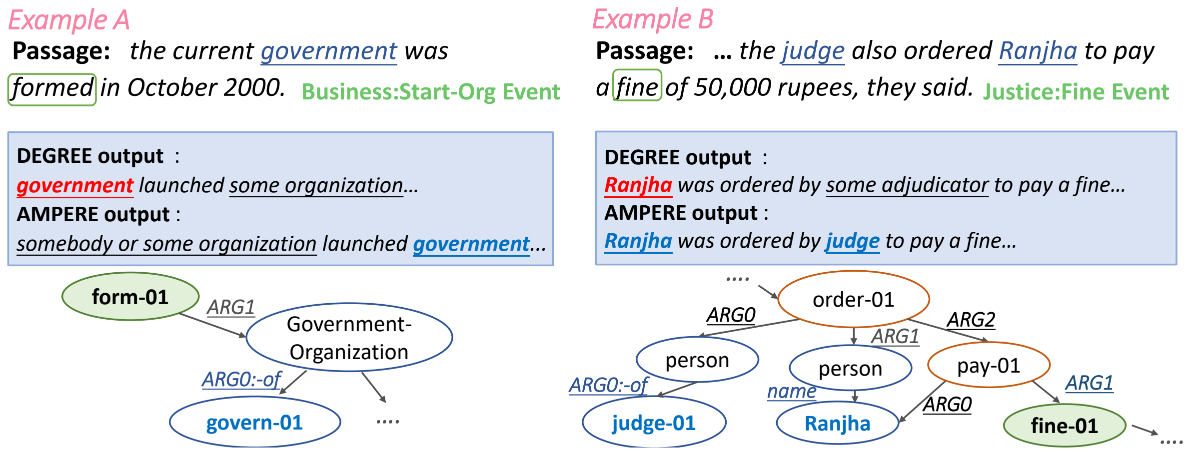

Figure 2: Two examples of how AMR information helps the generation of event argument predictions. Note that due to space constraints, the shown passage, output text, and AMR graph omit some irrelevant information.

The development of generation-based event (argument) extraction models starts from works investigating how to reformulate event extraction problems as a generation task (Du et al., 2021a,b). Follow-up works put efforts to show the influence of different prompt designs to the generative event models. (Ma et al., 2022; Yao et al., 2022; Hsu et al., 2022a) More recently, researchers start to improve this series of work by designing different model architectures (Du et al., 2022; Zhu et al., 2022). However, very few efforts have been put into the ways and the effectiveness of incorporating auxiliary syntactic and semantic information into such models, even though this information has been shown to be beneficial in classification-based models. Hence, in this paper, we present the study and explore ways to incorporate this additional information for generation-based event models.

Improving event extraction with weakly-supervisions. Being a challenging task that requires deep natural language understanding to solve, many prior efforts have been put into investigating which auxiliary upstream task information is useful for event predictions. (Xu and Huang, 2022; Liu et al., 2019a; Huang et al., 2018; Veyseh et al., 2020) Liu et al. (2019a); Ahmad et al. (2021) leverages dependency syntactic structures of the input sentence to help cross-lingual event predictions. Huang et al. (2016, 2018) uses the similarity between AMR and event structures to perform zero-shot event extraction. More recently, Zhang and Ji (2021); Veyseh et al. (2020); Xu et al. (2022) investigates different message passing methods on AMR graph to help learn better representations for final classifications. Despite many efforts that have been put into the community, these methods are designed for classification-based models. This highlights the open area for research — how and whether incorporating such auxiliary information can also be helpful. We take a step forward in this direction and present AMPERE to showcase the possibility to improve the generation-based event models by such way.

6 Conclusion

In this paper, we present AMPERE, a generation-based model equipped with AMR-aware prefixes. Through our comprehensive studies, we show that prefixes can serve as an effective medium to connect AMR information and the space of generative models, hence achieving effective integration of the auxiliary semantic information to the model. Additionally, we introduce an adjusted copy mechanism to help AMPERE more accurately and stably generate output disregarding the additional noise brought from the AMR graph. Through our experiments, we show that AMPERE achieves consistent improvements in every setting, and the improvement is particularly obvious in low-resource settings.

Acknowledgments

We thank anonymous reviewers for their helpful feedback. We thank the UCLA PLUSLab and UCLA-NLP group members for their initial review and feedback for an earlier version of the paper. This research was supported in part by AFOSR MURI via Grant #FA9550-22-1-0380, Defense Advanced Research Project Agency (DARPA) via Grant #HR00112290103/HR0011260656, the Intelligence Advanced Research Projects Activity

(IARPA) via Contract No. 2019-19051600007, National Science Foundation (NSF) via Award No. 2200274, and a research award sponsored by CISCO.

Limitations

Our goal is to demonstrate the potential of incorporating AMR to improve generation-based EAE models. Although we have shown the strength of our method, there are still some limitations. First, our proposed techniques are based on the AMR graph generated by pre-trained AMR parsers. The generated AMR graphs inevitably have a certain possibility of being not perfect. Hence, the error propagation issues would happen to AMPERE. We hypothesize this is one of the reasons why the improvement of AMPERE is not necessarily significant when data is abundant. Yet, through our experimental results, we still show the benefit of incorporating this information, especially in the case of low-resource settings. Second, although our AMR-aware prefix design should be agnostic to the used generation-based EAE model, in our experiment, we only set DEGREE as our base generation-based EAE model. We leave the investigation on the generalizability of our AMR-prefix method to other base models as future work.

Ethics Considerations

Our method relies on a pre-trained AMR parser, which is built using pre-trained large language models (`AMRBART` & `RoBERTa`). It is known that the models trained with a large text corpus may capture the bias reflecting the training data. Therefore, it is possible that the AMR graph used in our method could contain certain biases. We suggest carefully examining the potential bias before applying AMPERE to any real-world applications.

References

Wasi Uddin Ahmad, Nanyun Peng, and Kai-Wei Chang. 2021. GATE: graph attention transformer encoder for cross-lingual relation and event extraction. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*.

Xuefeng Bai, Yulong Chen, and Yue Zhang. 2022. Graph pre-training for AMR parsing and generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse, LAW-ID@ACL 2013, August 8-9, 2013, Sofia, Bulgaria*.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D. Manning. 2014. Modeling biological processes for reading comprehension. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1499–1510, Doha, Qatar. Association for Computational Linguistics.

Michele Bevilacqua, Rexhina Blloshmi, and Roberto Navigli. 2021. One SPRING to rule them both: Symmetric AMR semantic parsing and generation without a complex pipeline. In *Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI)*.

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 of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*.

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, Sha Li, and Heng Ji. 2022. Dynamic global memory for document-level argument extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*.

Xinya Du, Alexander M. Rush, and Claire Cardie. 2021a. GRIT: generative role-filler transformers for document-level event entity extraction. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*.

Xinya Du, Alexander M. Rush, and Claire Cardie. 2021b. Template filling with generative transformers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*. Association for Computational Linguistics.

Steven Fincke, Shantanu Agarwal, Scott Miller, and Elizabeth Boschee. 2022. Language model priming for cross-lingual event extraction. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*.

Frederik Hogenboom, Flavius Frasincar, Uzay Kaymak, Franciska de Jong, and Emiel Caron. 2016. A survey of event extraction methods from text for decision support systems. *Decis. Support Syst.*

I-Hung Hsu, Kuan-Hao Huang, Elizabeth Boschee, Scott Miller, Prem Natarajan, Kai-Wei Chang, and Nanyun Peng. 2022a. 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, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*.

I-Hung Hsu, Kuan-Hao Huang, Shuning Zhang, Wenxin Cheng, Premkumar Natarajan, Kai-Wei Chang, and Nanyun Peng. 2022b. A simple and unified tagging model with priming for relational structure predictions. *arXiv preprint arXiv:2205.12585*.

Kuan-Hao Huang, I-Hung Hsu, Prem Natarajan, Kai-Wei Chang, and Nanyun Peng. 2022. Multilingual generative language models for zero-shot cross-lingual event argument extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*.

Kung-Hsiang Huang, Sam Tang, and Nanyun Peng. 2021. Document-level entity-based extraction as template generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5257–5269, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Lifu Huang, Taylor Cassidy, Xiaocheng Feng, Heng Ji, Clare R. Voss, Jiawei Han, and Avirup Sil. 2016. Liberal event extraction and event schema induction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Lifu Huang, Heng Ji, Kyunghyun Cho, Ido Dagan, Sebastian Riedel, and Clare R. Voss. 2018. Zero-shot transfer learning for event extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. 2022. Survey of hallucination in natural language generation. *arXiv preprint arXiv:2202.03629*.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*.

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, ACL 2020, Online, July 5-10, 2020*.

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, NAACL-HLT 2021, Online, June 6-11, 2021*.

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, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*.

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 (ACL)*.

Jian Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2019a. Neural cross-lingual event detection with minimal parallel resources. 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 2019, Hong Kong, China, November 3-7, 2019*.

Xiao Liu, Heyan Huang, Ge Shi, and Bo Wang. 2022a. Dynamic prefix-tuning for generative template-based event extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022b. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*.

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

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, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*.

Yaojie Lu, Qing Liu, Dai Dai, Xinyan Xiao, Hongyu Lin, Xianpei Han, Le Sun, and Hua Wu. 2022. Unified structure generation for universal information extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*.

Jie Ma, Shuai Wang, Rishita Anubhai, Miguel Ballesteros, and Yaser Al-Onaizan. 2020. Resource-enhanced neural model for event argument extraction. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*.

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 (Volume 1: Long Papers)*, pages 6759–6774, Dublin, Ireland. Association for Computational Linguistics.

Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Jie Ma, Alessandro Achille, Rishita Anubhai, Cícero Nogueira dos Santos, Bing Xiang, and Stefano Soatto. 2021. Structured prediction as translation between augmented natural languages. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*.

Tanmay Parekh, I-Hung Hsu, Kuan-Hao Huang, Kai-Wei Chang, and Nanyun Peng. 2022. GENEVA: pushing the limit of generalizability for event argument extraction with 100+ event types. *arXiv preprint arXiv:2205.12505*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*. Association for Computational Linguistics.

Zhiyi Song, Ann Bies, Stephanie M. Strassel, Tom Riese, Justin Mott, Joe Ellis, Jonathan Wright, Seth Kulick, Neville Ryant, and Xiaoyi Ma. 2015. From light to rich ERE: annotation of entities, relations, and events. In *Proceedings of the The 3rd Workshop on EVENTS: Definition, Detection, Coreference, and Representation, (EVENTS@HLP-NAACL)*.

Amir Pouran Ben Veyseh, Tuan Ngo Nguyen, and Thien Huu Nguyen. 2020. Graph transformer networks with syntactic and semantic structures for event argument extraction. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. 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)*.

Sijia Wang, Mo Yu, Shiyu Chang, Lichao Sun, and Lifu Huang. 2022. Query and extract: Refining event extraction as type-oriented binary decoding. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 169–182, Dublin, Ireland. Association for Computational Linguistics.

Xiaozhi Wang, Ziqi Wang, Xu Han, Zhiyuan Liu, Juanzi Li, Peng Li, Maosong Sun, Jie Zhou, and Xiang Ren. 2019. HMEAE: hierarchical modular event argument 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 2019, Hong Kong, China, November 3-7, 2019*.

Haoyang Wen, Ying Lin, Tuan Manh Lai, Xiaoman Pan, Sha Li, Xudong Lin, Ben Zhou, Manling Li, Haoyu Wang, Hongming Zhang, Xiaodong Yu, Alexander Dong, Zhenhailong Wang, Yi Ren Fung, Piyush Mishra, Qing Lyu, Dídac Surís, Brian Chen, Susan Windisch Brown, Martha Palmer, Chris Callison-Burch, Carl Vondrick, Jiawei Han, Dan Roth, Shih-Fu Chang, and Heng Ji. 2021. RESIN: A dockerized schema-guided cross-document cross-lingual cross-media information extraction and event tracking system. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstrations, NAACL-HLT*.

Xueqing Wu, Kung-Hsiang Huang, Yi R. Fung, and Heng Ji. 2022. Cross-document misinformation detection based on event graph reasoning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL.*

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, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*.

Zhiyang Xu and Lifu Huang. 2022. Improve event extraction via self-training with gradient guidance. *arXiv preprint arXiv:2205.12490*.

Hang Yan, Tao Gui, Junqi Dai, Qipeng Guo, Zheng Zhang, and Xipeng Qiu. 2021. A unified generative framework for various NER subtasks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*.

Yunzhi Yao, Shengyu Mao, Xiang Chen, Ningyu Zhang, Shumin Deng, and Huajun Chen. 2022. Schema-aware reference as prompt improves data-efficient relational triple and event extraction. *arXiv preprint arXiv:2210.10709*.

Hongming Zhang, Xin Liu, Haojie Pan, Yangqiu Song, and Cane Wing-Ki Leung. 2020. ASER: A large-scale eventuality knowledge graph. In *The Web Conference 2020 (WWW)*.

Zixuan Zhang and Heng Ji. 2021. Abstract meaning representation guided graph encoding and decoding for joint information extraction. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*.

Tong Zhu, Xiaoye Qu, Wenliang Chen, Zhefeng Wang, Baoxing Huai, Nicholas Jing Yuan, and Min Zhang. 2022. Efficient document-level event extraction via pseudo-trigger-aware pruned complete graph. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*.

A Datasets

We present detailed dataset statistics in Table Table 4.

B Implementation Details

This section introduces the implementation details for all the baseline models we use in this paper. Our experiments are run using our machine that equips 8 NVIDIA a6000 GPUs.

- **DyGIE++**: we use their official code to reimplement the model.[12] Their original model is built using `BERT` (Devlin et al., 2019). As we mentioned in Section 3.1, we replace the used pre-trained language model into `RoBERTa-large` and tune with learning rates $= \{1e-5, 2e-5, 3e-5\}$.

- **OneIE**: we use their official code[13] to train the model. Their original model is built using `BERT` (Devlin et al., 2019). As we mentioned in Section 3.1, we replace the used pre-trained language model into `RoBERTa-large` and tune with learning rates $= \{1e-5, 2e-5, 3e-5\}$.

- **Query and Extract**: we use their official code[14] to train argument detection model with learning rate $= 1e-5$, batch size $= 16$, training epoch $= 10$. Different from the official code, we used `RoBERTa-large` for pre-trained language model to ensure a fair comparison.

- **AMR-IE**: the original AMR-IE is an end-to-end event extraction model, so we adapt their official code[15] to event argument extraction task by giving gold triggers in model evaluation. We fixed pre-trained language model learning rate $= 1e-5$, then did hyperparameter searching from graph learning rate $= \{1e-3, 4e-3\}$ and batch size $= \{8, 16\}$.

- **PAIE**: we use their official code[16] to train the model with the default parameters for `BART-large`.

- **DEGREE**: we use their official code[17] to train the model with the default parameters for `BART-large`.

C Detailed Result

Table 5 shows the detailed results of our main experiments. We repeat running every experiment setting with three random seeds, and report their average Arg-I and Arg-C F1-scores, and the corresponding standard deviation scores.

[12] https://github.com/dwadden/dygiepp

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

[14] https://github.com/VT-NLP/Event_Query_Extract/

[15] https://github.com/zhangzx-uiuc/AMR-IE

[16] https://github.com/mayubo2333/PAIE/

[17] https://github.com/PlusLabNLP/DEGREE

Dataset	Split	#Docs	#Sents	#Events	#Event Types	#Args	#Arg Types
ACE05-E	Train (5%)	25	649	212	27	228	21
	Train (10%)	50	1688	412	28	461	21
	Train (20%)	110	3467	823	33	936	22
	Train (30%)	160	5429	1368	33	1621	22
	Train (50%)	260	8985	2114	33	2426	22
	Train (full)	529	17172	4202	33	4859	22
	Dev	28	923	450	21	605	22
	Test	40	832	403	31	576	20
ERE-EN	Train (5%)	20	701	437	31	640	21
	Train (10%)	40	1536	618	37	908	21
	Train (20%)	80	2848	1231	38	1656	21
	Train (30%)	120	4382	1843	38	2632	21
	Train (50%)	200	7690	3138	38	4441	21
	Train (full)	396	14736	6208	38	8924	21
	Dev	31	1209	525	34	730	21
	Test	31	1163	551	33	822	21

Table 4: Dataset statistics. "#Docs" means the number of documents; "#Sents" means the number of sentences, "#Events" means the number of events in total; "#Event Types" means the size of event types set; "#Args" means the number of argument in total; "#Arg Types" means the size of argument role types set.

ACE05-E Development Set

Model	5%		10%		20%		30%		50%		100%	
	Arg-I	Arg-C	Arg-I	Arg-C	Arg-I	Arg-C	Arg-I	Arg-C	Arg-I	Arg-C	Arg-I	Arg-C
DyGIE++ (Wadden et al., 2019)	44.6±2.28	34.6±1.83	57.3±0.91	48.5±0.35	58.9±1.53	52.5±0.85	63.0±2.05	57.5±1.34	65.4±0.49	57.9±0.59	67.2±1.78	60.0±0.35
OneIE (Lin et al., 2020)	48.0±2.27	38.6±1.11	62.3±0.61	56.0±1.01	68.2±0.84	63.2±1.16	73.0±1.20	67.6±0.42	74.6±0.60	70.4±0.46	76.0±1.95	71.8±1.54
Query and Extract (Wang et al., 2022)	41.6±2.50	10.5±0.82	43.0±2.10	27.7±1.00	49.0±3.00	37.6±0.66	58.8±1.83	50.0±1.52	61.7±3.33	54.6±3.58	67.9±1.86	61.7±2.67
AMR-IE (Zhang and Ji, 2021)	49.7±1.12	40.0±1.29	62.0±0.34	56.4±0.83	66.8±0.90	61.3±1.23	72.4±1.28	67.4±0.66	74.7±1.04	70.6±1.30	77.7±0.93	73.1±0.68
PAIE (Ma et al., 2022)	55.2±1.16	46.6±0.98	64.1±0.88	57.6±1.43	70.4±0.49	64.6±1.13	74.5±0.38	69.3±0.38	75.1±1.89	70.3±1.02	78.5±0.65	74.1±0.80
DEGREE (Hsu et al., 2022a)	47.6±0.64	41.4±0.50	65.1±0.75	56.8±0.50	69.7±0.50	62.5±0.55	75.6±0.43	68.9±0.54	75.9±0.57	70.5±0.28	78.4±0.38	73.8±0.58
AMPERE (AMRBART)	$\underline{59.9}$±1.99	$\underline{52.3}$±1.54	**68.5**±0.83	**61.5**±0.82	**72.0**±0.80	**67.2**±0.55	$\underline{76.5}$±1.01	$\underline{71.2}$±0.56	**76.5**±0.50	**72.7**±0.83	$\underline{80.0}$±1.06	$\underline{75.6}$±1.10
AMPERE (RoBERTa)	**62.1**±1.73	**53.2**±2.26	$\underline{68.2}$±0.39	**61.5**±1.24	$\underline{71.5}$±1.00	$\underline{66.6}$±0.12	**76.8**±0.37	**71.8**±0.53	$\underline{76.4}$±1.01	$\underline{72.5}$±0.79	**80.9**±0.60	**76.6**±0.78

ACE05-E Test Set

Model	5%		10%		20%		30%		50%		100%	
	Arg-I	Arg-C	Arg-I	Arg-C	Arg-I	Arg-C	Arg-I	Arg-C	Arg-I	Arg-C	Arg-I	Arg-C
DyGIE++ (Wadden et al., 2019)	39.2±4.20	29.3±2.63	50.5±1.44	42.2±0.85	57.7±1.11	49.5±0.75	59.9±0.97	53.2±1.38	61.0±2.62	54.4±1.10	63.6±1.74	57.4±1.87
OneIE (Lin et al., 2020)	41.3±1.97	34.6±1.88	55.4±2.29	50.0±1.51	64.6±2.54	59.6±1.12	67.8±1.50	63.0±1.43	72.0±0.43	68.3±0.92	73.7±0.87	70.7±0.38
Query and Extract (Wang et al., 2022)	36.8±3.44	11.0±0.50	33.1±5.45	20.9±2.83	45.6±0.93	34.3±1.30	51.1±3.78	44.3±4.01	56.1±4.87	49.6±5.15	62.4±2.10	59.1±1.88
AMR-IE (Zhang and Ji, 2021)	43.2±1.54	36.8±0.07	53.3±1.49	48.5±0.99	63.2±0.60	58.3±0.93	67.2±1.00	62.6±1.16	69.5±1.27	66.1±0.92	73.6±0.40	70.3±0.13
PAIE (Ma et al., 2022)	52.2±0.83	46.3±0.75	62.0±0.96	56.3±0.46	67.8±0.33	62.8±0.69	71.3±0.54	65.8±0.98	72.8±2.34	69.1±2.20	75.0±0.51	72.1±0.69
DEGREE (Hsu et al., 2022a)	47.7±0.09	41.7±0.83	63.0±1.45	57.7±1.72	64.2±0.57	58.9±1.00	70.3±1.16	65.8±1.50	71.4±0.26	68.2±0.25	75.6±0.79	73.0±0.53
AMPERE (AMRBART)	$\underline{59.8}$±2.00	$\underline{52.4}$±1.53	$\underline{66.0}$±1.82	$\underline{61.0}$±1.58	$\underline{70.2}$±0.84	**66.4**±1.04	**73.3**±0.45	**69.7**±0.41	$\underline{74.4}$±1.21	$\underline{71.1}$±1.17	$\underline{76.0}$±0.85	$\underline{73.4}$±0.58
AMPERE (RoBERTa)	**61.0**±0.98	**53.4**±0.21	**67.8**±1.13	**61.7**±0.79	**70.5**±1.28	**66.4**±0.86	$\underline{73.1}$±0.43	$\underline{69.5}$±0.67	**74.6**±1.03	**71.9**±0.89	**76.7**±0.75	**74.2**±0.28

ERE-EN Development Set

Model	5%		10%		20%		30%		50%		100%	
	Arg-I	Arg-C	Arg-I	Arg-C	Arg-I	Arg-C	Arg-I	Arg-C	Arg-I	Arg-C	Arg-I	Arg-C
DyGIE++ (Wadden et al., 2019)	51.8±2.16	42.2±0.68	52.9±3.19	45.4±2.65	56.8±1.84	49.0±0.58	57.3±0.67	50.1±0.96	58.8±0.56	51.5±1.47	63.8±2.20	56.8±1.93
OneIE (Lin et al., 2020)	56.8±3.14	51.4±2.58	65.6±0.42	59.5±0.71	68.6±0.55	62.0±0.56	70.8±0.75	65.5±0.51	73.6±0.56	68.5±0.67	75.5±0.26	71.2±0.13
Query and Extract (Wang et al., 2022)	34.8±6.37	22.0±5.30	45.9±1.59	37.3±2.03	49.0±4.31	41.2±3.38	56.1±1.32	49.4±1.19	63.5±1.81	57.0±1.34	70.4±2.17	65.0±2.16
AMR-IE (Zhang and Ji, 2021)	48.4±1.48	44.8±0.86	61.2±0.97	55.2±1.06	63.0±1.37	56.9±1.12	70.5±0.03	65.2±0.63	73.0±0.79	67.6±0.39	75.3±1.30	70.1±1.45
DEGREE (Hsu et al., 2022a)	64.2±0.33	57.2±0.21	69.7±0.36	62.5±0.89	69.2±0.42	63.9±0.55	73.4±0.35	67.1±0.11	75.4±0.52	70.2±0.48	77.4±0.32	73.3±0.52
AMPERE (AMRBART)	$\underline{69.2}$±1.64	$\underline{62.4}$±1.54	**72.8**±1.12	**66.8**±1.03	$\underline{71.5}$±0.51	$\underline{66.0}$±0.95	**74.9**±0.65	$\underline{68.8}$±0.17	**76.7**±0.33	**70.8**±0.55	**78.1**±0.69	$\underline{73.6}$±1.10
AMPERE (RoBERTa)	**69.9**±0.97	**63.1**±1.24	$\underline{72.7}$±0.81	$\underline{66.7}$±0.56	**71.7**±0.33	**66.6**±0.98	$\underline{74.6}$±0.52	**69.7**±0.68	$\underline{75.7}$±0.74	$\underline{70.6}$±0.67	$\underline{77.9}$±0.28	**73.8**±0.34

ERE-EN Test Set

Model	5%		10%		20%		30%		50%		100%	
	Arg-I	Arg-C	Arg-I	Arg-C	Arg-I	Arg-C	Arg-I	Arg-C	Arg-I	Arg-C	Arg-I	Arg-C
DyGIE++ (Wadden et al., 2019)	53.3±1.95	40.0±1.93	52.9±2.59	44.6±2.70	55.9±1.74	49.5±1.16	59.1±0.64	52.0±1.35	60.5±0.92	53.7±0.38	63.4±0.80	56.0±0.78
OneIE (Lin et al., 2020)	55.5±3.47	49.5±2.24	62.1±1.53	56.1±1.62	67.9±1.83	62.3±1.62	71.9±0.36	66.1±0.73	72.3±0.49	67.7±0.43	75.2±1.14	70.1±1.96
Query and Extract (Wang et al., 2022)	35.1±7.25	19.7±5.12	46.7±2.66	34.0±4.06	52.1±4.69	42.4±5.07	57.7±0.09	50.1±0.86	64.5±2.78	57.7±2.80	70.4±1.78	64.3±2.26
AMR-IE (Zhang and Ji, 2021)	47.8±0.65	44.1±0.46	59.1±0.96	53.7±0.58	65.8±1.68	60.4±1.22	71.4±1.31	65.7±1.45	73.9±0.44	68.8±0.29	76.5±1.20	71.5±1.34
DEGREE (Hsu et al., 2022a)	66.4±0.14	57.5±0.36	71.2±1.26	63.9±1.38	72.3±0.69	67.4±0.56	74.1±1.16	69.1±1.44	**77.4**±0.61	**73.3**±0.74	78.2±0.69	74.9±1.10
AMPERE (AMRBART)	$\underline{71.3}$±0.40	$\underline{62.9}$±0.53	$\underline{73.7}$±0.73	$\underline{66.7}$±0.45	$\underline{73.1}$±0.57	**68.5**±0.44	**75.7**±0.83	**71.3**±0.70	$\underline{77.1}$±0.30	$\underline{72.5}$±1.07	**78.8**±0.62	**75.4**±0.59
AMPERE (RoBERTa)	**71.4**±1.13	**63.2**±0.57	**73.8**±0.57	**67.7**±0.66	**73.6**±0.64	$\underline{68.4}$±0.40	$\underline{75.4}$±0.36	$\underline{70.5}$±0.17	77.0±0.73	$\underline{72.5}$±0.68	$\underline{78.4}$±0.80	$\underline{75.0}$±0.77

Table 5: Argument Identification and classification F1-scores (%) under different data proportion settings for ACE05-E and ERE-EN datasets. The highest scores are in bold and the second-best scores are underlined. The reported numbers are the average of the results from three random runs. The standard deviation (%) of three runs are also reported in the table.