Eliciting Knowledge from Language Models for Event Extraction

Jiaju Lin,1 Jin Jian, 2 Qin Chen 3
1 2 3 East China Normal University
jiaju_lin@stu.ecnu.edu.cn , jjin@cs.ecnu.edu.cn , qchen@cs.ecnu.edu.cn

Abstract

Eliciting knowledge contained in language models via prompt-based learning has shown great potential in many natural language processing tasks, such as text classification and generation. Whereas, the applications for more complex tasks such as event extraction are less studied, since the design of prompt is not straightforward due to the complicated types and arguments. In this paper, we explore to elicit the knowledge from pre-trained language models for event trigger detection and argument extraction. Specifically, we present various joint trigger/argument prompt methods, which can elicit more complementary knowledge by modeling the interactions between different triggers or arguments. The experimental results on the benchmark dataset, namely ACE2005, show the great advantages of our proposed approach. In particular, our approach is superior to the recent advanced methods in the few-shot scenario where only a few samples are used for training.

Introduction

Event extraction aims to transform text into structural records. For example, in Figure 1, the raw sentence would be transformed into the structural informative records shown in the table on the right side. The table-like records include event types, trigger words and corresponding arguments.

Mainstream methods can be roughly categorized into two groups: (1) The sequence tagging based approaches that treat the event extraction task as trigger/argument identification and classification. (Lin et al. 2020)(Wadden et al. 2019) (2) The generation based approaches that generate trigger and argument tokens given the event descriptions or event schema(Lu et al. 2021)(Li, Ji, and Han 2021). Recent studies focus on enhancing the performance with the pretrained language models (PLMs)(Du and Cardie 2020)?(?). In particular, QA based MRC models are proposed for event extraction, which design the related questions manually and provide the answer as the trigger/argument.

Despite these inspiring progress, current methods still struggle due to the following critical issues. (1) The knowledge contained in the pretrained language models is not fully exploited. (Lu et al. 2021) modifies event extraction as a generation task, and introduces a sequence-to-structure

Figure 1: An example of event extraction from ACE2005 corpus. There are three events happening in this sentence. While only one trigger word, ‘killed’, links to two arguments directly, other triggers’ arguments need to be inferred via the information provided by ‘killed’ event.

model to extract trigger and arguments jointly. But the event schema they used lacks informative notion, making the potential of PLMs can not be completely unleashed. (2) Mechanisms to capture the interactions among arguments are not elegant and optimal. (Xiangyu et al. 2021) attempted to model the relations among both inter and intra event arguments via an RNN model. Whereas, the performance was heavily dependent on named entity recognition, which was prone to error propagation. (Wei et al. 2021) introduced a teacher-student mechanism to distill the knowledge from various teacher models. Specifically, the teacher models were trained to predict argument given relevant roles of a specific event, while the student model was trained for learn the knowledge from teachers for argument prediction. To impart all-around knowledge for argument extraction, many well-trained teacher models are needed, which poses big challenge for fine-tuning.

To address the above issues, we propose a PrOmpt-based Knowledge Eliciting approach (PoKE), which elicits specific knowledge from PLMs for event extraction. Specifically, we first design trigger/argument prompts based on the annotation guidelines, and then ask the model to fill the blanks with trigger words or argument texts. In this way, the event extraction is transformed into a conditional generation task, where the knowledge obtained during pretraining are elicited to generate the answer. In order to better capture the relations between different events and arguments, we present various joint prompt strategies to gauge the complementary knowledge about triggers and arguments that occur in the
Our approach is superior to the existing methods in three folds. (1) We mingle description understanding and interaction modeling task together. Not only can we elicit knowledge embedded in PLM, but also making full use of the PLM’s understanding ability. For instance, it is challenging for most existing models to extract all information correctly from the sentence in Figure 1, where the arguments of three events overlap each other and some trigger-argument linkages are not direct. However, our model is able to infer the ‘place’ and ‘target’ argument of ‘suicide bombing’ with the help of trigger ‘killed’, capturing the interactions between triggers efficiently. (2) Different from current generation models that generate arguments according to a predefined schema, we generate arguments separately. Thus our results would not suffer from the sub-optimal order of generation. Moreover, our model can take related arguments into consideration with the joint prompt. (3) Instead of introducing extra modules, we simply adjust the format of input data and realize the purpose of capturing interactions among triggers and arguments. Such brief mechanism decreases the amount of model parameters and alleviates the difficulty of fine tuning.

We evaluate our approach on the benchmark event extraction dataset, namely ACE2005. Experimental results show that our approach outperforms the state-of-the-art methods for argument extraction. Besides we accomplish competitive performance with our event extraction pipeline. It is also notable that our approach is capable of better learning in the low resource set, and outperforms the recent advanced methods under the few-shot scenario.

Our contributions can be summarized as follows:

- As far as we know, it is the first attempt to explore the effectiveness of eliciting knowledge from PLMs for event extraction, where the prompt design is not straightforward due to the complicated types and arguments.
- We develop various joint prompt strategies, which can capture the interactions among different triggers, intra and inter-event arguments, without any redundant modules.
- Extensive experiments on the most widely used ACE-2005 event extraction dataset demonstrate the superiority of our approach under both fully supervised and few-shot learning scenarios.

The Proposed Approach

In this section, we first present an overview of our framework. Then, we introduce various prompt methods to elicit knowledge from PLMs for trigger identification and argument extraction. After that we describe the details of prompt-based event extraction.

Framework Overview

Our PoKE framework depends on the natural language understanding ability of T5. To be specific, the different prompts we design would make our model understand the purpose of our task, i.e. comprehend linkage between answer and given context, rather than remember the distribution of answers. For event detection task, we split it into two parts. First, we generate the trigger words corresponding to 8 main types in each sentence, i.e., Movement, Personnel, Conflict, Contact, Life, Transaction, Justice, Business. With that result, we further classify triggers into 33 subtypes, also model it as a generation task. For event argument extraction task, our model would fill the bank in each prompt with argument span from the sentence, with trigger words and event type provided.

Event Detection Owing to the diversity of event subtypes, we first extract triggers according to 8 main types, and then classify them into subtypes. For extracting triggers for main type, we develop two joint trigger prompts to elicit more related knowledge by modeling the event interactions.

External Joint Trigger Prompt. Given a sentence, we add 8 prompts of all main types after it, asking the model to generate the trigger words jointly. In this manner, model can detect the discrepancy between various event types. Intuitively, we use the event type as an inevitable part of the prompt. Besides, considering the fact that contexts may render cues for the event, we expand the boundary of sentence. With a window of size=100 words, our sentence is set at the middle position and includes contexts in the window into the final passage. To illuminate which sentence triggers should be extracted from, two special tokens ‘(e)’ and ‘(\(e\)’) are applied to mark the original boundary of the raw sentence. Furthermore, to make the paragraph more natural sounding, we insert a bridge section, ‘In the passage above’, between the passage and prompts. A typical trigger external joint trigger prompt is shown in Figure 2. If an event does not occur in the sentence, we use ‘None’ to represent the NA answer.

Internal Joint Trigger Prompt. Given the fact that relevant events occur close to each other in a paragraph of text, it is sensible to teach our model how to capture the interaction between triggers. Due to that reason, we present internal joint trigger prompt for trigger extraction. Events within one sentence has certain relation with each other. Besides, we assume that the events appearing within a span of three sentences have potential interdependence. Thus we filter ineligible samples and concatenate adjoin eligible sentences as raw sentences. Similar to trigger prompt, we employ a window with size = 120 words to incorporate the contextual information. Unlike external joint trigger prompt that the masks are outside of the raw sentence, the internal joint trigger prompt masks the trigger words inside the raw sentence as shown in Figure 2.

With the identified triggers for the main event type, we need to further classify them into 33 subtypes. We inherit the trigger prompts and make classifications among the subtypes. As shown in Figure 2 with the identified triggers ‘convicted’ and ‘sentenced’ for the main event type ‘Justice’, we perform fine-grained classifications to classify them into the subtypes of events, namely ‘Convivt’ and ‘Sentence’.

Argument Extraction After event detection, we need to extract the event related arguments for better understanding.
Abbas was convicted in absentia by an Italian court and sentenced to five life terms in prison.

Figure 2: Architecture of PoKE for coarse-grained event detection. It is the first part of event detection, in this step, we extract triggers for eight main types. For this purpose, we introduce external joint trigger prompt and internal joint trigger prompt. We develop the former by setting masks in the prompts, to teach model how to infer trigger via the combination of passage and prompts. To generate the latter, we mask the triggers in the passage, asking the model to fill the mask via the information in the prompts. During this process, our model would model the interactions among triggers.

For each role in an event, to incorporate more semantic information into our templates, we first utilize the descriptions in the ACE annotation guidelines for events. Particularly, we make a masked position in the argument description text and urge the model to generate the corresponding words. In the case where multiple entities in a passage represent the same argument, we insert ‘|’ in the gold answer to split them. In addition, it is necessary for the model to have the knowledge of the trigger and its type. To provide the trigger and event information, we stick the event type and the corresponding trigger at end of the sentence. Both the trigger word in the sentence and template is surrounded by a pair of special tokens as ⟨t⟩/⟨/t⟩. An example of the single argument prompt is presented in Figure 3.

Joint Argument Prompt. When trying to understand an event from a passage, it is natural for a human to infer the arguments from other co-occurred arguments. To stimulate such learning process, we present a joint argument prompt method to elicit more related knowledge by modeling the interactions among the intra-event arguments and inter-event arguments.

For each sentence that contains an event, we provide the corresponding arguments information behind it, while mask the arguments words in the sentence. The ACE annotation guidelines are applied to generate the arguments description information. Also, the trigger word and event type are added after the argument descriptions. Besides, to prevent our model from cheating by heuristic relations, like filling the blanks according to the order of arguments mentioned in prompts, we shuffle the list of prompts randomly. Figure 3 shows an example of our joint argument prompt. With this prompt, more complementary knowledge can be elicited from the pre-trained language models, building solid relations between the descriptions in prompts and the concrete argument role for more accurate argument extraction.

Prompt-based Event Extraction

Since our base model architecture is an encoder-decoder design. We employed T5 for the generation task. The generation process follows T5 pre-training task, a slew of extra tokens, from ⟨extra_id_0⟩ to ⟨extra_id_99⟩ are exploited as masks. We replace the specific word spans with these special tokens. And the generation output is the answers joined by these extra tokens, like ⟨extra_id_0⟩ Argument_0 ⟨extra_id_1⟩ Argument_1 ⟨extra_id_2⟩/⟨/s⟩, here ⟨/s⟩ means the end of the sentence.

The generation process models the conditional probability
of selecting a new token given the previous tokens and the input to the encoder.

\[
p(x|\text{prompt}) = \prod_{i=1}^{x} p(x_i|x_{i-1}, \text{prompt})
\] (1)

The model is trained by minimizing the cross entropy loss over vocabulary \(V\).

\[-\sum_{i=1}^{V} y_i \log(p_i(x|\text{prompt}))\] (2)

In inference stage, we use greedy decoding, i.e. choosing the highest-probability logit at every times step. Given the fact that the vocabulary space is huge and it is possible for the model to generate words out of the passage. We constrain the candidate vocabulary to \(V_s\), the set of tokens in the input, by setting logits of extraneous words to \(-\infty\).

\[
p = \begin{cases} p(x|\text{prompt}) & w \in V_s \\ -\infty & w \notin V_s \end{cases}
\] (3)

**Experiments**

**Experimental Setup**

**Data set.** We evaluate our work on the most widely used ACE2005 dataset, containing 599 documents annotated with 33 event subtypes and 35 argument types. For fair competition, we follow the same split and preprocessing step as the previous work (Yang et al. 2018) (Wadden et al. 2019) (Du and Cardie 2020). The data set splits are as follows, 40 newswire documents as test set, 30 randomly selected documents for development and a training set including the rest 529 documents.

**Baselines.** For complete event extraction task we compare our method with five baselines. Considering the difference in the annotation methods and label-intensity, we categorize them into three types following (Lu et al. 2021) 1. Baseline using token annotation and entity annotation: OneIE (Lin et al. 2020), an end-to-end system, employs global features to conduct named entity recognition, relation extraction and event extraction at the same time.

2. Baselines using token annotation: CondiGen (Li, Ji, and Han 2021) is one of the cutting-edge method for argument extraction task. It utilizes the conditionally generation ability of BART. Given description of arguments and their relations then generate the corresponding words in the sentence. The candidate arguments are predicted jointly. However, it gives up generation model and introduces a modified TapNet for event detection, formulating it as classic sequence tagging task. EEQA which is one of the earliest work that introduces a new paradigm for event extraction by formulating it as a question answering task. It generates questions from annotation guidelines and raise question for each argument.
Table 1: Experiment results of event extraction on ACE2005. Trig-C indicates trigger identification and classification. Arg-C indicates argument identification and classification. The column PLM indicates the pre-trained language models used by each method.

Table 2: Results of event argument extraction. PoKE single only means we train the model only with single argument prompts. PoKE w.o. inter indicates we replace joint argument prompts with the single prompts but mask in the passage, which means no interaction information is impart to the model.

Supervised Learning Results

Event Extraction Results

The performance of PoKE and all baselines on ACE05 are presented in Table 1 where ‘Trig-C’ indicates trigger identification and classification and ‘Arg-C’ reflects the result of argument identification and classification. We can observe that by leveraging interactions among triggers and arguments and taking advantage of prompts, our model achieve comparable or better performance with the baseline models. Compared with the other two global-generation model, TANL and Text2Event, our method overtake them both in event detection and argument extraction task. Notice that the results of argument extraction is directly affected by the output of event detection. An error prediction of event type would render wrong candidate classes for argument extractor. Hence the reduction in F1 after argument extraction indicates our merits in argument extraction.

Argument Extraction Results

Since the current comparisons are based on the trigger predictions of previous event detection work (Wang et al. 2019a). It maybe reasonable when no advance can be achieved for event detection task. However, with the rapid evolution in event detection methods (Pouran Ben Veyseh et al. 2021), we believe such previous criterion can not show the whole picture of each architecture’s ability. Hence, following the hyper parameter settings mentioned in their paper, we reproduce the results of several cutting-edge models with gold triggers as input to do a fair comparison. Meanwhile, we introduce another baseline developed for event argument extraction specifically.

HMEAE which applies a concept hierarchy indicating argument roles. With the help of hierarchical modular attention, correlation between argument roles sharing the same high-level unit can be utilized. (Wang et al. 2019b). It should be noticed that HMEAE takes use of the named entity information as candidate arguments. Also, we do some ablation study to examine the efficiency of our model. ‘PoKE single only’ means we train the model only with single argument prompts. ‘PoKE w.o. inter’ indicates we replace joint argument prompts with the single prompts but mask in the passage, which means no interaction information is impart to the model.

The results are shown in Table 2 from which we can observe that (1) Our method surpasses all baseline methods, with an absolute improvement of at least 1.3 in F1. (2) Models which consider the interactions between arguments perform better. For instance, CondiGen is trained to generate all arguments of a single event jointly. Although the manual templates used for its inference are sub-optimal, it overtakes EEQA with 3.1 in F1. Taking advantages of both joint prediction and informative description, our model achieves the state-of-the-art performance. (3) Named entity annotation is not an absolute

| Model       | P    | R    | F1 | Encoder/PLM | PLM  | Annotation    |
|-------------|------|------|----|-------------|------|---------------|
| PoKE        | 64.1 | 76.2 | 69.6 | T5-base     |      | Token         |
| PoKE w.o. inter | 61.5 | 67.1 | 64.2 | T5-base     |      | Token         |
| PoKE single only | 60.7 | 64.9 | 62.7 | 2*BERT      |      | Token         |

3. Baseline using Text-Record Annotation: Text2Event, an end-to-end generation-based, modeling event-detection and argument extraction tasks in a single model. Compared with normal generation methods, it outputs a structural event record with event type, trigger, and arguments (Lu et al. 2021).

Implementation

We optimized our model using AdamW (Kalchbrenner and Blunsom 2013) with different hyperparameters for each task. For coarse-grained event detection, we set learning rate=1e-4 training epoch=6. For fine-grained trigger classification, learning rate=5e-4 batch size=8 training epoch=3. For argument extraction task, learning rate=1e-3 and batch size=64. The experiments are conducted on a single NVIDIA GeForce RTX 3090. The random seed set for all tasks is 42.
necessity. Although without the assistance of named entity recognition results, models utilize PLMs achieve competitive performances. Particularly, input of HMEAE are sentences with named entity annotation, however, its encoder, a vanilla CNN, restricts its ability. (4) Joint argument prompts enhance model’s performance dramatically, with almost 8 in F1, from 62.79 to 70.10.

![Figure 4: Few-shot experiment results. K-shot corresponds to the number of data in training and validation set.](image)

**Table 3: Compare PoKE with Data Augment Methods**

| Method          | P    | R    | F1   |
|-----------------|------|------|------|
| EDA             | 54.46| 77.43| 63.94|
| BackTranslation | 54.03| 76.95| 63.48|
| PoKE single     | 60.79| 64.92| 62.79|
| PoKE            | 66.20| 74.48| 70.10|

Few Shot Scenario

Imitating few-shot settings in LM-BFF (Gao, Fisch, and Chen 2021), we conduct 8,16,32 shot experiments to examine the few-shot ability of our model. Specifically, for each event type, we sample k instances from initial training and validation set. After training, we test the model performance on standard scale test set. The experiment results are in Table 2. It can be observed that our model perform better than baselines in few-shot scenario. Although EEQA and CondiGen both have outstanding zero-shot ability, when it comes to few-shot task, they are exceeded by our PoKE. Even a 32-shot trained EEQA model can not overtake our PoKE trained by 8-shot data. Our mechanism leverages naturalness into prompts which ensure its few-shot performance.

Compared with Data Augment Methods

PoKE increases the number and variety of training samples, which would enhance model’s ability naturally, while data augment methods can improve model’s performance in the same way. Hence we compare our method with two most common seen data augment methods, back translation and EDA (Wei and Zou 2019), which consists of four word-level operations, replacement, insertion, swap, and deletion, to boost diversity of sentences. Experiments are conducted on argument extraction task. We discard the interaction learning data for two data augment methods and expand the argument inference data set to the same number of training data used for PoKE. The results are demonstrated in Table 3. From the result, we can see that compared with using inference data only, data augment methods do improve model’s performance. Nonetheless, the improvement is limited. It further demonstrates that our mechanism unleash PLM’s understanding ability.

Case Study and Error Analysis

Case Study

To demonstrate how our model utilize information from co-occurring events, we show extraction results of samples with multi events in Figure 5. Sentence S1 contains two trigger words: ‘destroyed’ and ‘killed’, indicating ‘Die’ and ‘Attack’ event respectively. It is a common fact that the agent responsible for a death is usually the attacking agent in an attack. Also, word ‘commandos’ is the subject of the sentence, strongly syntactically related to both two triggers. Given that, our model is capable of exploiting the clues hidden behind and makes the correct prediction of ‘Die-Agent’ and ‘Attack_Attacker’. Besides, although there is a false prediction of ‘Victim’, i.e., our model identifies ‘soldiers’ as victims while no ‘Victim’ is annotated. We believe it is the gold answer that contains a mistake. In many cases, our model makes more sensible argument extraction that human annotated data.

Error Analysis

To facilitate further research, we investigate the error predictions of our model. We randomly sampled 50 wrong answers and attributed them to 5 types. Here we list 3 most common problems caused by our model design.

Shallow Heuristics obtained in training set(24%) Data in the training set always contains some bias that induce heuristic relations between entity mentions and corresponding argument types. Model learns these heuristic relations to enhance its performance, while neglect the intended task. We define the event argument extraction task as utilizing the sentence s, trigger t, event type e to infer corresponding arguments a: \{s, t, e\} → \{a\}. However, we observed that, in some sense, extraction of arguments relies on the entity mention in the sentence and event type, \{s, e\} → \{a\}, omitting the relation between entity and trigger word. For example, in the training set, when an event about ‘Justice’ and entity ‘court’ appear in the same sentence, ‘court’ always plays a role of ‘Adjudicator’. Thus, when the same situation occurs in the inference period, our model identifies ‘court’ in the sentence as ‘Adjudicator’, no matter whether it is linked to the trigger word.

One of the error prediction is as follows:

The Belgrade district [court] Adjudicator said that Markovic will be tried along with 10 other Milosevic-era officials who face similar (t0)charges/(t0) of inappropriate use of state property that carry a sentence of up to five years in jail.

In this instance, ‘court’ has no direct relation to the trigger word. Thus the gold annotation of ‘Adjudicator’ is
Australian commandos, who have been operating deep in Iraq, destroyed a command and control post and killed a number of soldiers, according to the country’s defense chief, Gen. Peter #CARDIE2020.

Figure 5: Case study of PoKE and EEQA

‘None’. However, due to the wrong experience accumulated in training stage, our model classifies it as a adjudicator because the sentence contains a Justice-Sentence event.

Ambiguous expression (20%) Owing to the ambiguity of natural language, even human reader would make some mistakes, especially under the situation where no context are given. Ambiguous expression error refers to the case when our model makes a prediction inconsistent with gold answer, but reasonable from some perspectives. These mistakes usually occur when inferring ‘Place’ arguments or arguments expressed implicitly. For example, in the sentence ‘Kelly, the US assistant secretary for East Asia and Pacific Affairs, arrived in Seoul from Beijing Friday to brief Gen. Yoon, the foreign minister.‘ Seoul is predicted as the ‘Place’ of ‘Contact.Meet’ event, but no ‘Place’ argument is annotated. It is true that the place of meeting is not mentioned clearly, but for humans, applying rhetoric to imply implicit arguments is not rare sense.

We also find the gold sample expressing place argument implicitly: ⟨t0⟩Former President Carol Moseley Braun recently decided to run for president. Here ‘Illinois’ is annotated as the place where employment relationship ends.

Coreference & head of entity mistake(10%) Errors ascribed to this kind refers to the situation when the model predicts a coreferent of gold annotation or the whole entity span rather than the head of it. For example, “[Erdogan] person, a leader of Turkey’s pro-Islamic movement when he was jailed, said he moderated his policies in prison.” ‘Erdogan’ is predicted as ‘Person’ argument of ‘Justice.Arrest-Jail’ event, while the gold annotation is ‘leader’. For head of entity mistake, the model fails to figure out the head word of entity span correctly. Here is a concrete example: “Anwar will be taken to the [t0] appeal court early Friday for a bail application pending his appeal to the country’s highest [FederalCourt] Adjudicator against his sodomy conviction, counsel Sankara Nair said.” The gold answer of ‘Adjudicator’ is ‘Court’ but ‘Federal Court’ is predicted by our model.

Related Work

Event extraction

Most efforts in event extraction field are concentrated on the ACE2005 corpus. Various of neural models have been exploited for better performance. (Nguyen and Grishman 2015) first introduced Convolutional Neural Network, (CNN) for event detection for better representation of features. And (Nguyen, Cho, and Grishman 2016) employed recurrent neural network to build robust model. Besides, Graph Convolutional Neural Networks (Liu, Luo, and Huang 2018) and plenty of fancy ideas have been applied to handle difficulties we meet (Lu et al. 2021). Recently, there is a trend of modeling event extraction into QA-based tasks (Du and Cardie 2020). These researches employ reading comprehension abilities of natural language models. Despite these advantages, current methods still suffer from the bottlenecks mentioned in Introduction.

Prompt-based learning

Prompt-based learning is based on language models that model the probability of text directly. Prompt is a piece of string that contains descriptive information about the answer of the task. The language model is used to probabilistically fill the unfilled slot to obtain a final string as the prediction of the task. Exploration in prompt methods can be categorized into prompt engineering which refers to creating a prompting function $f_p(x)$ that results in efficient performance on the downstream task (Shin et al. 2020), answering engineering which targets at looking for an optimal answer space and a map to the original output (Shin et al. 2020), and multi-prompt learning which involves prompt ensemble (Yuan, Neubig, and Liu 2021), prompt augmentation (Gao, Fisch, and Chen 2021), prompt composition (Han et al. 2021) and prompt decomposition (Cui et al. 2021). In this work, we utilize manual prompts and take the idea of prompt ensemble, but only modify the format of prompts in training stage to increase the PLM’s understanding of the task.

Conclusion and Future Work

We propose a prompt-based approach (PoKE), which includes various trigger/argument prompts to elicit knowledge from language models for event extraction. In particular, the prompts model the interactions between different events and arguments, which gauges more complementary knowledge and yields better performance. The experimental results demonstrate the superiority of our approach in achieving comparable performance without additional model parameters or modules. It is also notable that we can still perform well under that few-shot scenario. In the future, we will introduce automatic prompt methods for this task and explore the effectiveness of prompt for joint event extraction.
References

Cui, L.; Wu, Y.; Liu, J.; Yang, S.; and Zhang, Y. 2021. Template-Based Named Entity Recognition Using BART. *CoRR*, abs/2106.01760.

Du, X.; and Cardie, C. 2020. Event Extraction by Answering (Almost) Natural Questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 671–683. Online: Association for Computational Linguistics.

Gao, T.; Fisch, A.; and Chen, D. 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 (Volume 1: Long Papers)*, 3816–3830. Online: Association for Computational Linguistics.

Han, X.; Zhao, W.; Ding, N.; Liu, Z.; and Sun, M. 2021. PTR: Prompt Tuning with Rules for Text Classification. *CoRR*, abs/2105.11259.

Jiang, Z.; Xu, F. F.; Araki, J.; and Neubig, G. 2020. How Can We Know What Language Models Know? *Transactions of the Association for Computational Linguistics*, 8: 423–438.

Kalchbrenner, N.; and Blunsom, P. 2013. Recurrent Continuous Translation Models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 1700–1709. Seattle, Washington, USA: Association for Computational Linguistics.

Li, S.; Ji, H.; and Han, J. 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*, 894–908. Online: Association for Computational Linguistics.

Lin, Y.; Ji, H.; Huang, F.; and Wu, L. 2020. A Joint Neural Model for Information Extraction with Global Features. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 7999–8009. Online: Association for Computational Linguistics.

Liu, X.; Luo, Z.; and Huang, H. 2018. Jointly Multiple Events Extraction via Attention-based Graph Information Aggregation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 1247–1256. Brussels, Belgium: Association for Computational Linguistics.

Lu, Y.; Lin, H.; Xu, J.; Han, X.; Tang, J.; Li, A.; Sun, L.; Liao, M.; and Chen, S. 2021. Texi2Event: Controllable Sequence-to-Structure Generation for End-to-end Event Extraction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2795–2806. Online: Association for Computational Linguistics.

Nguyen, T. H.; Cho, K.; and Grishman, R. 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*, 300–309. San Diego, California: Association for Computational Linguistics.

Nguyen, T. H.; and Grishman, R. 2015. Event Detection and Domain Adaptation with Convolutional Neural Networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, 365–371. Beijing, China: Association for Computational Linguistics.

Paolini, G.; Athiwaratkun, B.; Krone, J.; Ma, J.; Achille, A.; Anubhai, R.; dos Santos, C. N.; Xiang, B.; and Soatto, S. 2021. Structured Prediction as Translation between Augmented Natural Languages. In *9th International Conference on Learning Representations, ICLR 2021*.

Pouran Ben Veyseh, A.; Lai, V.; Dernoncourt, F.; and Nguyen, T. H. 2021. Unleash GPT-2 Power for Event Detection. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 6271–6282. Online: Association for Computational Linguistics.

Shin, T.; Razeghi, Y.; Logan IV, R. L.; Wallace, E.; and Singh, S. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 4222–4235. Online: Association for Computational Linguistics.

Wadden, D.; Wennberg, U.; Luan, Y.; and Hajishirzi, H. 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)*, 5784–5789. Hong Kong, China: Association for Computational Linguistics.

Wang, X.; Han, X.; Liu, Z.; Sun, M.; and Li, P. 2019a. Adversarial Training for Weakly Supervised Event Detection. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 998–1008. Minneapolis, Minnesota: Association for Computational Linguistics.

Wang, X.; Wang, Z.; Han, X.; Liu, Z.; Li, J.; Li, P.; Sun, M.; Zhou, J.; and Ren, X. 2019b. 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)*, 5777–5783. Hong Kong, China: Association for Computational Linguistics.

Wei, J.; and Zou, K. 2019. EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks. 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)*, 6382–6388. Hong Kong, China: Association for Computational Linguistics.

Wei, K.; Sun, X.; Zhang, Z.; Zhang, J.; Zhi, G.; and Jin, L. 2021. Trigger is Not Sufficient: Exploiting Frame-aware Knowledge for Implicit Event Argument Extraction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 4672–4682. Online: Association for Computational Linguistics.

Xiangyu, X.; Ye, W.; Zhang, S.; Wang, Q.; Jiang, H.; and Wu, W. 2021. Capturing Event Argument Interaction via A Bi-Directional Entity-Level Recurrent Decoder. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 210–219. Online: Association for Computational Linguistics.

Yang, P.; Sun, X.; Li, W.; Ma, S.; Wu, W.; and Wang, H. 2018. SGM: Sequence Generation Model for Multi-label Classification. In *Proceedings of the 27th International Conference on Computational Linguistics*, 3915–3926. Santa Fe, New Mexico, USA: Association for Computational Linguistics.

Yuan, W.; Neubig, G.; and Liu, P. 2021. BARTScore: Evaluating Generated Text as Text Generation. *arXiv:2106.11520*. 

