

Resource-Enhanced Neural Model for Event Argument Extraction

Jie Ma* Shuai Wang*

Rishita Anubhai Miguel Ballesteros Yaser Al-Onaizan

Amazon AI

{jieman, wshui, ranubhai, ballemig, onaizan}@amazon.com

Abstract

Event argument extraction (EAE) aims to identify the arguments of an event and classify the roles that those arguments play. Despite great efforts made in prior work, there remain many challenges: (1) Data scarcity. (2) Capturing the long-range dependency, specifically, the connection between an event trigger and a distant event argument. (3) Integrating event trigger information into candidate argument representation. For (1), we explore using unlabeled data in different ways. For (2), we propose to use a syntax-attending Transformer that can utilize dependency parses to guide the attention mechanism. For (3), we propose a trigger-aware sequence encoder with several types of trigger-dependent sequence representations. We also support argument extraction either from text annotated with gold entities or from plain text. Experiments on the English ACE2005 benchmark show that our approach achieves a new state-of-the-art.

1 Introduction

Event argument extraction (EAE) aims to identify the entities that serve as arguments of an event and to classify the specific roles they play. As in Fig. 1, “two soldiers” and “yesterday” are *arguments*, where the event triggers are “attacked” (with event type being ATTACK¹) and “injured” (event type INJURY). For the trigger “attacked”, “two soldiers” plays the argument role *Target* while “yesterday” plays the argument role *Attack Time*. For the event trigger “injured”, “two soldiers” and “yesterday” play the role *Victim* and *INJURY Time*, respectively. There has been significant work on event extraction (EE) (Liao and Grishman, 2010; Hong et al., 2011; Li et al., 2013), but the EAE

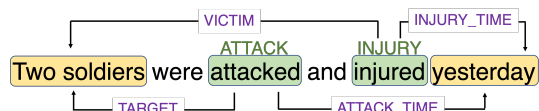


Figure 1: Event examples: Green indicates triggers with their types. Yellow indicates arguments. An arrow indicates the role played by an argument in a trigger.

task remains a challenge and has become the bottleneck for improving the overall performance of EE (Wang et al., 2019a).²

Supervised data for EAE is expensive and hence scarce. One possible solution is to use other available resources like unlabeled data. For that, (1) We use BERT (Devlin et al., 2018) as our model encoder which leverages a much larger unannotated corpus where semantic information is captured. Unlike Yang et al. (2019) who added a final/prediction layer to BERT for argument extraction, we use BERT as token embedder and build a sequence of EAE task-specific components (Sec. 2). (2) We use (unlabeled) in-domain data to adapt the BERT model parameters in a subsequent pretraining step as in (Gururangan et al., 2020). This makes the encoder domain-aware. (3) We perform self-training to construct auto-labeled data (*silver data*).

A crucial aspect for EAE is to integrate event trigger information into the learned representations. This is important because arguments are dependent on triggers, i.e., the same argument span plays completely different roles toward different triggers. An example is shown in Fig. 1, where “two soldiers” plays the role *Target* for the event ATTACK and the role *Victim* for INJURY. Different from

²EAE has similarities with semantic role labeling. Event triggers are comparable to predicates in SRL and the roles in most SRL datasets have a standard convention of interpreting *who* did *what* to *whom*. EAE has a custom taxonomy of roles by domain. We also use inspiration from the SRL body of work (Strubell et al., 2018; Wang et al., 2019b; He et al., 2017; Marcheggiani and Titov, 2017).

*Indicates Equal Contribution.

¹Following ACE <https://www ldc.upenn.edu/collaborations/past-projects/ace>

existing work that relies on regular sequence encoders, we design a novel trigger-aware encoder which simultaneously learns four different types of trigger-informed sequence representations.

Capturing the long-range dependency is another important factor, e.g., the connection between an event trigger and a distant argument. Syntactic information could be useful in this case, as it could help bridge the gap from a word to another distant but highly related word (Sha et al., 2018; Liu et al., 2018; Strubell et al., 2018). We modify a Transformer (Devlin et al., 2018) by explicitly incorporating syntax via an attention layer driven by the dependency parse of the sequence.

We design our role-specific argument decoder to seamlessly accommodate both settings (with and without the availability of entities). We also tackle the *role overlap problem* (Yang et al., 2019) using a set of classifiers or taggers in our decoder.

Our model achieves the new state-of-the-art on ACE2005 Events data (Grishman et al., 2005).

2 Event Argument Extraction

2.1 Task Setup

Consider a sequence $\mathcal{X} = \{x_1, \dots, x_i, \dots, x_T\}$ of T tokens x_t . A span $x_{ij} = \{x_i \dots x_j\}$ is a subsequence in \mathcal{X} . An event trigger g is a span x_{ab} indicating an event of type y_g , where y_g belongs to a fixed set of pre-defined trigger types. Given a sequence-trigger pair (\mathcal{X}, g) as input, EAE has two goals: (1) Identify all *argument* spans from \mathcal{X} and (2) Classify the *role* r for each argument. In some settings, a set of entities is given (each entity is a span in X) and such entities are used as a candidate pool for arguments. For example, “two soldiers” and “yesterday” are candidate entities in Fig. 1.

2.2 Modeling Argument Extraction

Fig. 2 presents our model architecture with the following components: (1) trigger-aware sequence encoder, (2) syntax-attending Transformer and (3) role-specific argument decoder.

Trigger-Aware Sequence Encoder: This encoder is designed to distinguish candidate arguments conditioned on different triggers. Note a span may encode different argument information for two triggers, for example, in Fig. 1, “Two soldiers” plays the role of *Target* for the ATTACK event and *Victim* for the INJURY event. In order to model this, our encoder uses BERT to embed input

tokens, where the BERT embedding of token x_t is denoted as b_t . A segment (0/1) embedding seg_t for each token x_t indicating whether x_t belongs to the trigger or not (Logeswaran et al., 2019, inter-alia) is used, which is added up with token embedding and position embedding as input to BERT (Fig.2). The encoder then concatenates the following learned representations³ for each token: (1) A trigger representation h_g by max pooling over BERT embeddings of the tokens in trigger g ; (2) A trigger type embedding p_{y_g} for y_g ; (3) A trigger indicator (0/1) embedding l_t , indicating whether x_t belongs to the trigger or not.⁴ (4) A token embedding b_t . This results in a trigger-aware representation c_t for each token where $c_t = Concat(b_t; p_{y_g}; l_t; h_g)$ and C for the whole sequence with T tokens.

Syntax-Attending Transformer: Dependency parsing has been used as a feature to improve EE (Sha et al., 2018; Liu et al., 2018). Inspired by Strubell et al. (2018), we utilize dependency parses⁵ by modifying an attention head for each layer in a Transformer. Note that this Transformer is different from the BERT component, as this Transformer aims to capture long-range dependency on top of the trigger-aware representations learned from our sequence encoder. The output C from our encoder now will be the input of this Transformer, which will go through L layers of the modified syntax-attending Transformers. Each of these is assumed to have N self-attention heads. For each layer l , we modify one of these N heads to be a dependency-based attention head (call d-head) with output H^l :

$$H^l = W_2([W_1 U^{(l-1)}; A^l]), \quad (1)$$

where

$$A^l = Attention(W_Q^l Q^l, W_K^l K^l, W_V^l V^l). \quad (2)$$

Q^l, K^l and V^l are query, key, and value representations, and W_* are learning parameters. $U^l = \{u_1^l \dots u_T^l\}$ is the layer- l output of our Transformer and $U^0 = C$. Eq. 2 uses the scaled-dot product attention (Vaswani et al., 2017). The difference of the

³(2) and (3) are randomly initialized. They will be jointly learned during model training.

⁴While both l_t and seg_t are used for indicating whether a token x_t belongs to a trigger, the difference is that l_t is used to encode such information after obtaining BERT outputs and seg_t is used as an input to BERT. They are of different sizes.

⁵We use Stanford Parser <https://nlp.stanford.edu/software/lex-parser.shtml>

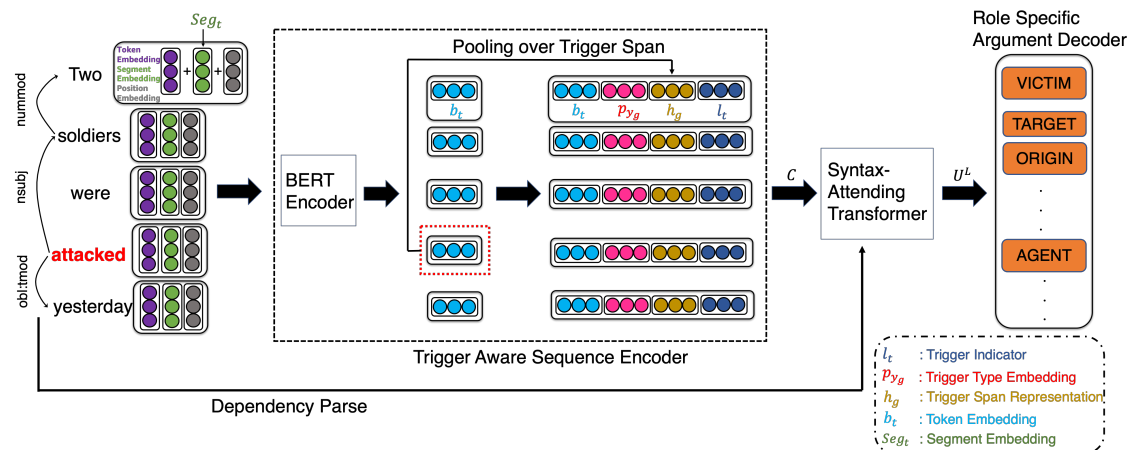


Figure 2: Model Architecture

d-head compared to other heads is that its keys K and values V are constructed differently. For each token x_i , valid keys and values are restricted to all tokens x_j such that x_i and x_j have an edge between them⁶ in the dependency parse of the sequence \mathcal{X} . This makes every $a_t^l \in A^l$ a weighted attention sum over the neighbor⁷ values v_j^l of the token x_i in the dependency parse. We then concatenate this a_t^l and the token’s own representation u_t^l projected linearly. Finally, this is projected back to the same dimensions as the outputs of the other $N - 1$ attention heads. By concatenating their outputs, our model captures both syntax-informed and global-attending information. The final output from our Transformer component is $U^L = u_1^L, u_2^L \dots u_T^L$.

Role-Specific Argument Decoder: We consider two settings: (1) with and (2) without entities. When entities are provided, they are used to form candidates for arguments; when they are not provided, our model infers arguments from plain text.

For (1), we assume that all arguments are entities but the vice versa is not true. So, we treat all entity spans, within a fixed sentence window around the trigger g , as candidate arguments. An entity representation is formed by pooling u_t^L for all tokens x_t in the entity span. Note that, since the encoder is trigger-aware, this representation is already conditioned on (\mathcal{X}, g) for role classification. Commonly used datasets like ACE2005 have a 10% *role overlap problem* (Yang et al., 2019). Concretely, consider a sentence like “The suicide bomber died in the blast he set off”. Here, “sui-

⁶ x_i may represent a subword unit. We assume all subwords of a word in the dependency parse inherit the head and children from the parent word.

⁷Head and children.

cide bomber” plays two distinct roles *Attacker* and *Victim* for the same trigger “blast” that denotes an ATTACK event. Hence, we perform role classification for every role independently (as a multi-label classification problem), using a set of classifiers, where each classifier handles one particular role, i.e., role-specific (such as the *VICTIM*, *TARGET* or *ORIGIN* as orange shown in Fig. 2). We thus call this decoder role-specific argument decoder.

More specifically, we use one binary classifier per role permissible for current trigger type on this entity representation. The outcome of the classifier for role r determines whether this entity plays the role r for the current trigger or not.

For (2), in the absence of entities we have no candidate spans for arguments. Using final layer output of syntax-attending Transformer, we predict a sequence of BIO tags with one sequence tagger per role.⁸ So in this setting the role-specific argument decoder comprises a set of sequence taggers.

2.3 Training Regimes for Data Scarcity

Domain-adaptive pretraining: An additional phase of in-domain pretraining has been shown to be effective for downstream tasks (Gururangan et al., 2020). Based on this, we perform a second phase of domain-adaptive pretraining with both BERT losses before fine-tuning the BERT encoder.

Self-training: For self-training (Chapelle et al., 2009; Scudder, 1965, inter-alia), we first train our model based on gold data. Next, we use that model

⁸Each token is tagged with B, I or O, indicating a token is at the beginning, inside of, or outside an argument span. Here we replace the conventional multi-class BIO tagger with a set of role-wise taggers. So tokens play role r have the tags B and I from the role-specific tagger for r .

to tag unlabeled data and get a much larger but noisy silver dataset (Sec. 3). We then train a new version of our model on the silver dataset; the resulting model is later fine-tuned on the gold data.

Auxiliary tasks: Although trigger detection is not the focus of this work, we model it as an auxiliary task to help EAE. We share the BERT encoder (Sec. 2.2) for both tasks. The trigger detection task uses the standard sequence tagging decoder for BERT (Devlin et al., 2018). The intuition here is to improve (1) the representation of the shared BERT component and (2) trigger representation, by performing trigger detection.

3 Experiments

Data and Tools: We use the ACE-2005 English Event data (Grishman et al., 2005).⁹ For self-training and domain-adaptive pretraining, we randomly sample 50k documents (626k sentences) from Gigaword¹⁰ to construct silver data. We use Stanford CoreNLP software¹¹ for tokenization, sentence segmentation and dependency parsing.

Training Setup: We use 50 dimensions for trigger indicator and trigger type embedding. We use 2 ($L = 2$) layers for the syntax-attending Transformer with 2 ($N = 2$) attention heads, dropout of 0.1. When entities are available, we only consider entities in the same sentence as the trigger as candidates for argument extraction. During training, We use Adam (Kingma and Ba, 2014) as optimizer and batch size of 32 for both main task EAE and the auxiliary task of trigger detection; we alternate between batches of main task and auxiliary task with probabilities of 0.9 and 0.1, respectively. We early stop training if performance on the development set does not improve after 20 epochs. All model parameters are fine-tuned during training. For BERT pretraining, we use the same setting as in (Devlin et al., 2018) but with an initial learning rate of $1e-5$. We stop pretraining after 10k steps. In order to obtain reliably predicted triggers as input for EAE, we trained a five-model ensemble trigger detection system following Wadden et al. (2019).¹²

⁹Standard splits (Li et al., 2013): 529 documents (14,385 sentences) are used for training, 30 documents (813 sentences) for development, and 40 documents (632 sentences) for test.

¹⁰<https://catalog.ldc.upenn.edu/LDC2011T07>

¹¹<https://stanfordnlp.github.io/CoreNLP/>

¹²Since trigger detection is not our main task and improving it is not the focus of this work, its results are not for comparison and thus excluded from the main result tables. As a

Model	Argument Identification (AI)			Role Classification (RC)		
	P	R	F1	P	R	F1
(Yubo et al., 2015)	68.8	51.9	59.1	62.2	46.9	53.5
(Nguyen et al., 2016)	61.4	64.2	62.8	54.2	56.7	55.4
(Sha et al., 2016)	63.2	59.4	61.2	54.1	53.5	53.8
(Sha et al., 2018)	71.3	64.5	67.7	66.2	52.8	58.7
(Yang et al., 2019)	71.4	60.1	65.3	62.3	54.2	58.0
(Wang et al., 2019a)	-	-	-	62.2	56.6	59.3
(Liu et al., 2018)	71.4	65.6	68.4	66.8	54.9	60.3
Ours	64.8	63.7	64.2	61.1	60.6	60.8
Ours + Pretraining	65.8	62.9	64.3	62.3	60.0	61.1
Ours + Self Training	64.5	65.0	64.7	61.1	62.3	61.7
(Sha et al., 2018) [†]	-	-	57.2	-	-	50.1
(Zhang et al., 2019) [†]	63.3	48.7	55.1	61.6	45.7	52.4
(Nguyen and Nguyen, 2019) [†]	59.9	59.8	59.9	52.1	52.1	52.1
(Wadden et al., 2019) [†]	-	-	55.4	-	-	52.5
(Zhang et al., 2020) [†]	-	-	-	54.5	52.4	53.4
Ours [†]	55.6	57.9	56.7	53.0	55.7	54.3
Ours + Pretraining [†]	56.3	58.1	57.2	53.5	55.8	54.6
Ours + Self Training [†]	58.4	56.9	57.6	56.0	54.8	55.3

Table 1: Experimental results. † indicates a model does not use gold entities. Ours show the mean of 5 random seeds. P refers to precision and R refers to recall.

Results and Analyses: Table 1 shows the results in two experimental settings: with and without entities. In the setting with entities, we use gold entities as in prior work. We have the following observations: (1) Our model achieves the best results ever reported in both experimental settings on RC (overall F1 scores). (2) Our model does not achieve the highest scores on AI. It seems however that our model is able to bridge the gap given that in order to achieve good results in RC you need AI, so it couples these two mutually affected sub-tasks closer to each other. (3) Self-training leads to gains of 1 F1 point both with and without entities. (4) Domain-adaptive pretraining shows small improvements on both AI and RC (less than self-training). There are two possible reasons. First, Gigaword is news while ACE is not only news; we only adapted part of the domains. Second, even though we used a small learning rate during pretraining, 50k unlabeled documents is a small amount for pretraining.

Ablation Study: We ablate each of the components and show results in Table 2. We observe that (1) All components help. We can see the performance gain of each component in the settings of with and without entities. (2) The overall trigger-aware sequence encoder leads to ~ 1.5 F1 points gain in both settings. (3) The use of the auxiliary task and the syntax-encoder improve by another ~ 1 F1 points.

reference, our five-model ensemble achieves 73.88 F1 score in the trigger classification task on ACE2005 Event test set.

Model	w/ entities	w/o entities
Argument (Single Task)	68.1	62.1
Argument + TI	68.5	62.5
Argument + TI + TT	69.1	63.3
Argument + TI + TT + TE	69.6	63.5
Arg. + Tri. (Auxiliary)	70.2	64.2
Arg. + Tri. (Auxiliary) + Syntax.	70.8	64.6

Table 2: Ablation analysis of our model on development set with gold trigger. TT = Trigger Type. TI = Trigger Indicator. TE = Trigger Embedding. Tri. = Trigger. Arg. = Argument + TI + TT + TE. Auxiliary indicates that trigger detection here is an auxiliary task. w/ and w/o entities mean with and with out entities provided. Results are the mean over 5 random seeds.

4 Related Work

Event Argument Exaction (EAE) is an important task in Event Extraction (EE). Early studies designed lexical, contextual or syntactical features to tackle the EE problem (Ji and Grishman, 2008; Liao and Grishman, 2010; Hong et al., 2011; Li et al., 2013). Later on, neural networks (Yubo et al., 2015; Sha et al., 2016; Nguyen et al., 2016; Sha et al., 2018; Liu et al., 2018; Yang et al., 2019; Wang et al., 2019a) demonstrated their effectiveness in representation learning without manual feature engineering. Our proposed model belongs to the latter category.

Here we present and discuss the most related studies to our work. Yang et al. (2019) used a pre-trained model with a state-machine based span boundary detector. They used heuristics to resolve final span boundaries. Wang et al. (2019a) also used a pre-trained model together with a hand-crafted conceptual hierarchy. Our approach does not need the design of such heuristics or conceptual hierarchy. In terms of modeling, their approaches used regular BERT as their encoders, where the argument representations are not explicitly conditioned on triggers. In contrast, our encoder is enhanced by providing more trigger-oriented information and BERT is only used as one part of it, which results in a trigger-aware sequence encoder. This allows us to better model interactions between arguments and triggers. Liu et al. (2018) added a GCN layer to integrate the syntactic information into a neural model. Different from their solution, we encode the syntax jointly with attention mechanism, simplifying the learning, making it more efficient, and achieving better results. Finally, no prior work has deeply studied the data scarcity is-

sue in EAE, while we exploit several techniques to tackle it in this work.

5 Conclusion

We present a new model which provides the best results in the EAE task. The model can generate trigger-aware argument representations, incorporate syntactic information (via dependency parses), and handle the role overlapping problem with role-specific argument decoder. We also experiment with some methods to address the data scarcity issue. Experimental results show the effectiveness of our proposed approaches.

Acknowledgments

We would like to thank the anonymous reviewers for their comments and suggestions.

References

- Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. 2009. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Ralph Grishman, David Westbrook, and Adam Meyers. 2005. Nys english ace 2005 system description. *ACE*, 5.
- Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don’t stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*.
- Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what’s next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 473–483, Vancouver, Canada. Association for Computational Linguistics.
- Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1127–1136. Association for Computational Linguistics.
- Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of ACL-08: Hlt*, pages 254–262.

- Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 73–82.
- Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 789–797. Association for Computational Linguistics.
- Xiao Liu, Zhunchen Luo, and Heyan Huang. 2018. Jointly multiple events extraction via attention-based graph information aggregation. *arXiv preprint arXiv:1809.09078*.
- Lajanugen Logeswaran, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, Jacob Devlin, and Honglak Lee. 2019. [Zero-shot entity linking by reading entity descriptions](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3449–3460, Florence, Italy. Association for Computational Linguistics.
- Diego Marcheggiani and Ivan Titov. 2017. [Encoding sentences with graph convolutional networks for semantic role labeling](#). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515, Copenhagen, Denmark. Association for Computational Linguistics.
- Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 300–309.
- Trung Minh Nguyen and Thien Huu Nguyen. 2019. One for all: Neural joint modeling of entities and events. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6851–6858.
- H Scudder. 1965. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 11(3):363–371.
- Lei Sha, Jing Liu, Chin-Yew Lin, Sujian Li, Baobao Chang, and Zhifang Sui. 2016. Rbp: Regularization-based pattern balancing method for event extraction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1224–1234.
- Lei Sha, Feng Qian, Baobao Chang, and Zhifang Sui. 2018. Jointly extracting event triggers and arguments by dependency-bridge rnn and tensor-based argument interaction. In *Thirty-Second AAAI Conference on Artificial Intelligence*.
- Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. *arXiv preprint arXiv:1804.08199*.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.
- David Wadden, Ulme Wennberg, Yi Luan, and Hananeh 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)*, pages 5788–5793.
- Xiaozhi Wang, Ziqi Wang, Xu Han, Zhiyuan Liu, Juanzi Li, Peng Li, Maosong Sun, Jie Zhou, and Xiang Ren. 2019a. 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)*, pages 5781–5787.
- Yufei Wang, Mark Johnson, Stephen Wan, Yifang Sun, and Wei Wang. 2019b. [How to best use syntax in semantic role labelling](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5338–5343, Florence, Italy. Association for Computational Linguistics.
- Sen Yang, Dawei Feng, Linbo Qiao, Zhigang Kan, and Dongsheng Li. 2019. Exploring pre-trained language models for event extraction and generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5284–5294.
- Chen Yubo, Xu Liheng, Liu Kang, Zeng Daojian, Zhao Jun, et al. 2015. Event extraction via dynamic multi-pooling convolutional neural networks.
- Tongtao Zhang, Heng Ji, and Avirup Sil. 2019. Joint entity and event extraction with generative adversarial imitation learning. *Data Intelligence*, 1(2):99–120.
- Yunyan Zhang, Guangluan Xu, Yang Wang, Daoyu Lin, Feng Li, Chenglong Wu, Jingyuan Zhang, and Tinglei Huang. 2020. A question answering-based framework for one-step event argument extraction. *IEEE Access*, 8:65420–65431.