EventGraph: Event Extraction as Semantic Graph Parsing

Huiling You,[1] **David Samuel,**[1] **Samia Touileb,**[2] and **Lilja Øvrelid**[1]
[1]University of Oslo
[2]University of Bergen
`{huiliny, davisamu, liljao}@ifi.uio.no`
`samia.touileb@uib.no`

Abstract

Event extraction involves the detection and extraction of both the event triggers and corresponding event arguments. Existing systems often decompose event extraction into multiple subtasks, without considering their possible interactions. In this paper, we propose Event-Graph, a joint framework for event extraction, which encodes events as graphs. We represent event triggers and arguments as nodes in a semantic graph. Event extraction therefore becomes a graph parsing problem, which provides the following advantages: 1) performing event detection and argument extraction jointly; 2) detecting and extracting multiple events from a piece of text; and 3) capturing the complicated interaction between event arguments and triggers. Experimental results on ACE2005 show that our model is competitive to state-of-the-art systems and has substantially improved the results on argument extraction. Additionally, we create two new datasets from ACE2005 where we keep the entire text spans for event arguments, instead of just the head word(s). Our code and models are released as open-source.[1]

1   Introduction

Event extraction aims at extracting event-related information from unstructured texts into structured form (i.e. triggers and arguments), according to a predefined event ontology (Ahn, 2006; Doddington et al., 2004). In these types of ontologies, events are characterized by event triggers, and comprise a set of predefined argument types. Figure 1 shows an example of a sentence containing two events, an `Attack` event triggered by *"friendly-fire"* and a `Die` event triggered by *"died"*; the two events share the same arguments, but each plays a different role in the specific event. For instance, *"U.S."* is the `Agent` in the `Die` event, but plays the role of `Attacker` in the `Attack` event.
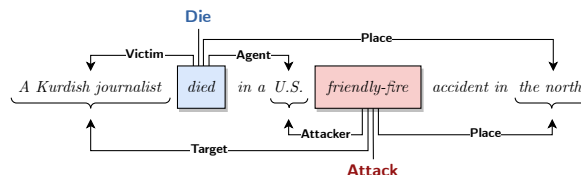

Figure 1: Example of an `Attack` and a `Die` events in the sentence "*A Kurdish journalist died in a U.S. friendly-fire accident in the north.*"

As opposed to dividing event extraction into independent subtasks, we take advantage of recent advances in semantic dependency parsing (Dozat and Manning, 2018; Samuel and Straka, 2020) and develop an end-to-end event graph parser, dubbed EventGraph. We adopt intuitive graph encoding to represent the event mentions of a piece of text in a single event graph, and directly generate these event graphs from raw texts. We evaluate our Event-Graph system on ACE2005 (LDC2006T06).[2] Our model achieves competitive results with state-of-the-art models, and substantially improves the results on event argument extraction. The main contributions of this work are:

1. We propose EventGraph, a text-to-event framework that solves event extraction as semantic graph parsing. The model does not rely on any language-specific features or event-specific ontology, so it can easily be applied to new languages and new datasets.

2. We design an intuitive graph encoding approach to represent event structures in a single event graph.

3. The versatility of our approach allows for an effortless decoding of full trigger and argument mentions. We create two novel and more challenging datasets from ACE2005, and provide corresponding benchmark results.

---

[1] https://github.com/huiling-y/EventGraph

[2] https://catalog.ldc.upenn.edu/LDC2006T06

2 Related work

Our work is closely related to two research directions, event extraction and semantic parsing.

Supervised event extraction is an established research area in NLP. There are different methods to obtain the structured information of an event, and the mainstream methods can be divided into: 1) classification-based methods: treat event extraction as several classification subtasks, and either solve them separately in a pipeline-based manner (Ji and Grishman, 2008; Li et al., 2013; Liu et al., 2020; Du and Cardie, 2020; Li et al., 2020) or jointly infer multiple subtasks (Yang and Mitchell, 2016; Nguyen et al., 2016; Liu et al., 2018; Wadden et al., 2019; Lin et al., 2020); 2) generation-based approaches: formulate event extraction as a sequence generation problem (Paolini et al., 2021; Lu et al., 2021; Li et al., 2021; Hsu et al., 2022); 3) prompt tuning methods: inspired by natural language understanding tasks, these approaches take advantage of "discrete prompts" (Shin et al., 2020; Gao et al., 2021; Li and Liang, 2021; Liu et al., 2022).

Meaning Representation Parsing has seen significant interest in recent years (Oepen et al., 2014, 2015, 2020). Unlike syntactic dependency representations, these semantic representations are crucially not trees, but rather general graphs, characterised by potentially having multiple entry points (*roots*) and not necessarily being connected, since not every token is a node in the graph. There has further been considerable progress in developing variants of both transition-based and graph-based dependency parsers capable of producing such semantic graphs (Hershcovich et al., 2017; Dozat and Manning, 2018; Samuel and Straka, 2020).

A recent and highly relevant development in the current context has been the application of semantic parsers to NLP tasks beyond meaning representation parsing. These approaches rely on the reformulation of task-specific representations to semantic dependency graphs. For example, Yu et al. (2020) exploit the parser of Dozat and Manning (2018) to predict spans of named entities, while Kurtz et al. (2020) phrase the task of negation resolution (Morante and Daelemans, 2012) as a graph parsing task with promising results. Recently, Barnes et al. (2021) proposed a dependency parsing approach to extract opinion tuples from text, dubbed Structured Sentiment Analysis, and a recent shared task dedicated to this task demonstrated the usefulness of graph parsing approaches to sentiment analysis
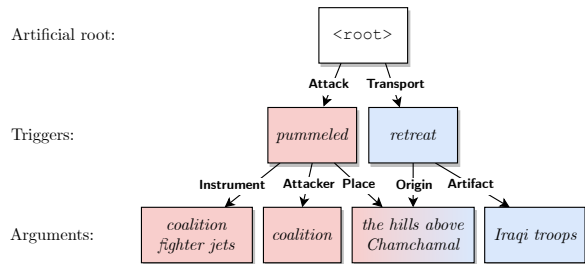

Figure 2: Event graph for the sentence *"That's because coalition fighter jets pummeled this Iraqi position on the hills above Chamchamal and Iraqi troops made a hasty retreat."*

(Barnes et al., 2022). Most similar to our work is the work by Samuel et al. (2022) which adapts the PERIN parser (Samuel and Straka, 2020) to parse directly from raw text into sentiment graphs.

3 Event graph representations

We adopt an efficient "labeled-edge" representation for event graph encoding within the scope of a sentence. Each node in an event graph corresponds to either an event trigger or an argument, which is anchored to a unique text span in a sentence, except for the top node, which is only a dummy node for every event graph. The edges are constrained only between the top node and an event trigger, or between an event trigger and an argument, with the corresponding edge label as an event type or argument role. The "labeled-edge" encoding has the ability to represent: 1) multiple event mentions; 2) nested structures (overlapping between arguments or trigger-argument); 3) multiple argument roles of a single argument. Taking the event graph from Figure 2 as example, the sentence contains two event mentions, which share the same argument *"the hills above Chamchamal"* but as different roles, and the argument *"coalition"* is nested inside the argument *"coalition fighter jets"*.

4 Event parsing

EventGraph is an adaptation of PERIN (Samuel and Straka, 2020), a general permutation-invariant framework for text-to-graph parsing. Given the "labeled-edge" encoding for event graphs, we create EventGraph by customizing the modules of PERIN as illustrated in Figure 3, which contains three classifiers to generate nodes, anchors, and edges, respectively. Each input sequence is processed by four modules of EventGraph to generate a final structured representation.
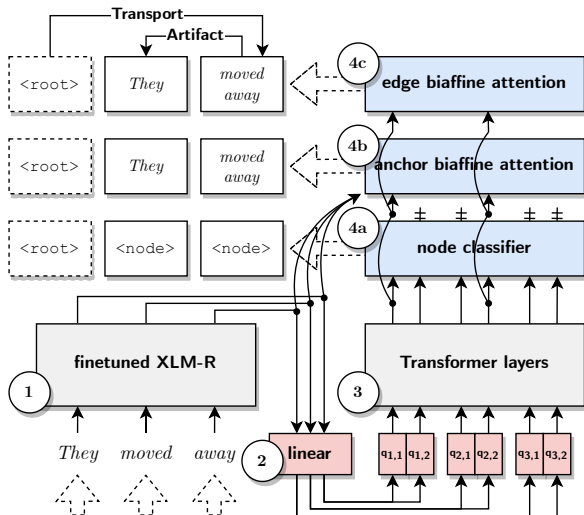
Figure 3: EventGraph architecture. 1) the input gets a contextualized representation, 2) queries are generated for every input token, 3) queries are further processed with a decoder to predict 4a) node presence, 4b) node anchors, and 4c) edge labels.

Encoder We use the large version of XLM-R (Conneau et al., 2020) as the encoder to obtain contextualized representations of the input sequence; each token gets a contextual embedding via a learned subword attention layer over the subwords.

Query generator We use a linear transformation layer to map each embebbed token onto n queries.

Decoder The decoder is a stack of Transformer encoder layers (Vaswani et al., 2017) without positional encoding, which is permutation-invariant (non-autoregressive); the decoder processes and augments the queries of each token by modelling the inter-dependencies between queries.

Parser head It consists of three classifiers: a) the **node classifier** is a linear classifier that predicts node presence by classifying the augmented queries of each token; since more than one query is generated for each token, a single token can produce more than one node; b) the **anchor biaffine classifier** (Dozat and Manning, 2017) uses deep biaffine attention between the augmented queries and contextual embeddings of each token to map the predicted nodes to surface tokens; c) the **edge biaffine classifier** uses two deep biaffine attention modules to process generated nodes and predict edge presence between a pair of nodes and the edge label.

Given a piece of text, EventGraph generates its corresponding graph, and it is effortless to extract the structured information of event mentions from the nodes and edges.[3]

5 Experimental setup

5.1 Datasets

We evaluate our system on the widely used benchmark dataset ACE2005[4] (LDC2006T06). The ACE2005 dataset contains 599 English documents annotated for several tasks, entities, values, relations, and events, with an event ontology of 33 event types, and 35 argument roles. Event arguments come from both entities and values. The annotation of an entity also includes its head word(s); for instance, from Table 1, entity "the Iraqi government's key center of power" has "center" as its head word. Following previous works (Wadden et al., 2019; Lin et al., 2020; Wang et al., 2019), we preprocess the dataset (details in Appendix B) and obtain the following configurations:

1. **ACE05-E**: Wadden et al. (2019) keep 22 event argument roles (excluding "time" and "value" event arguments), ignore events with multi-token trigger(s), and use only the head word(s) of event arguments.

2. **ACE05-E$^+$**: similar to Wadden et al. (2019), Lin et al. (2020) only use 22 event argument roles and keep only the head word(s) of event arguments, but keep events with multi-token trigger(s).

3. **ACE05-E^{++}**: we create a new dataset that keeps the full text spans for event triggers and event arguments, but also keep 22 argument roles for comparing with previous work.

4. **ACE05-E^{+++}**: we create another dataset that keeps all the 35 argument roles in ACE2005, with full text spans for event triggers and arguments.

Table 1 shows how an event mention is extracted in ACE05-E$^+$ and ACE05-E^{++}, and the same event is not present in ACE05-E. Although keeping the full text spans for arguments makes the task of argument extraction more difficult, we believe that the extracted events are more informative and self-contained.

---

[3]The tool for conversion between event mentions and event graphs is included in our codes.

[4]https://catalog.ldc.upenn.edu/LDC2006T06

	ACE05-E$^+$	ACE05-E^{++}
Trigger	*"push ahead"*	*"push ahead"*
Destination	*"center"*	*"the Iraqi government's key center of power"*
Artifact	*"forces"*	*"American forces"*

Table 1: A `Transport` event in "*Well, as American forces do push ahead toward the Iraqi government's key center of power, British forces are keeping up their work to the south of the Iraqi capital*", and corresponding extracted events in ACE05-E$^+$ and ACE05-E^{++}.

Dataset	Split	# Sentences	# Events	# Roles
	Train	17 172	4 202	4 859
ACE05-E	Dev	923	450	605
	Test	832	403	576
	Train	19 216	4 419	6 607
ACE05-E$^+$	Dev	901	468	759
	Test	676	424	689
	Train	15 603	4 416	6 513
ACE05-E^{++}	Dev	893	509	802
	Test	729	424	685
	Train	15 603	4 416	7 844
ACE05-E^{+++}	Dev	893	509	945
	Test	729	424	894

Table 2: Statistics of the preprocessed ACE2005 datasets.

Dataset	Triggers Avg. Len	Arguments Avg. Len	Single-token	Multi-token
ACE05-E	1.00	1.18	86.2%	13.8%
ACE05-E$^+$	1.06	1.17	88.0%	12.0%
ACE05-E^{++}	1.05	2.86	43.5%	56.5%
ACE05-E^{+++}	1.05	2.82	43.2%	56.8%

Table 3: Statistics of event triggers and arguments. We report the average lengths of triggers and arguments; for arguments, we also report the percentages of single-token and multi-token arguments.

5.2 Evaluation metric

We report Precision *P*, Recall *R*, and *F1* scores for each of the following evaluation criteria (Wadden et al., 2019; Lin et al., 2020):

- **Trigger classification (Trg-C)**: an event trigger is correctly predicted if its offsets and event type matches the gold trigger.

- **Argument classification (Arg-C)**: an event argument is correctly predicted if its offsets, argument role, and event type match the gold argument.

For argument classification, in order to have a better insight into our models' performance on multi-token arguments, we include another met-

ric based on token-level span overlap for argument identification, instead of perfect match.

- **Token-level span overlap**: an event argument is correctly identified if its offsets have 80%[5] overlap (token-level) with the gold argument, and correctly predicted if its argument role and event type match the gold argument.

5.3 System comparisons

We compare EventGraph to the following event extraction systems: 1) DYGIE++ (Wadden et al., 2019): a span-based framework capturing both local and global contexts; 2) ONEIE (Lin et al., 2020): an end-to-end framework for general information extraction; 3) TEXT2EVENT (Lu et al., 2021): a generation-based model for sequence-to-event generation; 4) GTEE-DYNPREF (Liu et al., 2022): a template-based method for text-to-event generation.

5.4 Implementation details

Our code is built upon the official implementation of the PERIN parser (Samuel and Straka, 2020).[6] Details about our training setup and hyperparameter settings are given in Appendix A. For each dataset, we train 5 models with 5 different random seeds, and report the means and standard deviations of the corresponding results.

6 Results and discussion

In Table 4, we compare our results on ACE05-E and ACE05-E$^+$ with the previous systems. On both datasets, EventGraph achieves SOTA results on Arg-C over all metrics, with an improvement of 7 percentage points on ACE-E and more than 10 percentage points on ACE05-E$^+$ in *F1* scores. For Trg-C, despite not beating the SOTA systems, our results are still very competitive.

On the two new datasets that we created, EventGraph has achieved overall competitive results (Table 4). On ACE-E^{++}, despite having longer and more complicated arguments, EventGraph has generated comparable results to those of GTEE-DYNPREF (current SOTA) on ACE-E$^+$. On ACE-E^{+++}, even though the argument role set is expanded from 22 to 35 argument roles, the results of EventGraph on Arg-C remain stable.

---

[5]This metric only affects arguments longer than 5 tokens. Arguments containing fewer than 5 tokens are still evaluated with perfect match.

[6]https://github.com/ufal/perin

Model	Triggers (Trg-C)			Arguments (Arg-C)		
	P	R	F1	P	R	F1
Dataset: ACE05-E						
DYGIE++	——	——	69.7	——	——	48.8
ONEIE	——	——	74.7	——	——	56.8
GTEE-DYNPREF	63.7	**84.4**	72.6	49.0	64.8	55.8
EventGraph	**66.5**$^{\pm0.7}$	71.0$^{\pm0.9}$	68.6$^{\pm0.7}$	**63.4**$^{\pm2.7}$	**67.3**$^{\pm2.0}$	**65.3**$^{\pm2.2}$
Dataset: ACE05-E$^+$						
ONEIE	**72.1**	73.6	72.8	55.4	54.3	54.8
TEXT2EVENT	71.2	72.5	71.8	54.0	54.8	54.4
GTEE-DYNPREF	67.3	**83.0**	**74.3**	49.8	60.7	54.7
EventGraph	70.0$^{\pm1.1}$	70.0$^{\pm1.2}$	70.0$^{\pm1.1}$	**64.5**$^{\pm1.0}$	**66.4**$^{\pm2.6}$	**65.4**$^{\pm1.7}$
Dataset: ACE05-E^{++}						
EventGraph	72.9$^{\pm1.3}$	75.2$^{\pm1.9}$	74.0$^{\pm1.5}$	57.3$^{\pm0.8}$	59.9$^{\pm1.2}$	58.6$^{\pm0.9}$
Dataset: ACE05-E^{+++}						
EventGraph	72.4$^{\pm0.7}$	75.9$^{\pm1.0}$	74.0$^{\pm0.7}$	56.9$^{\pm0.6}$	58.2$^{\pm0.9}$	57.5$^{\pm0.6}$

Table 4: Results on ACE05-E, ACE05-E$^+$, ACE05-E^{++}, and ACE05-E^{+++}. We report the average performance of 5 runs with different random seeds, together with the standard deviations. For clarity, we bold the highest scores.

Results show that EventGraph performs well on joint modelling of event triggers and arguments, and benefits from longer text spans for event triggers and arguments. When the full text spans of arguments are used, the model receives more training signals, so it has more information in differentiating sentences containing events from those without, as shown in Table 6, and thus identifying event triggers, which is also shown by the increasing Trg-C scores from ACE-E and ACE-E$^+$ to ACE-E^{++} and ACE-E^{+++}. For instance, as the example in Table 1 shows, "the Iraqi government's key center of power" is less ambiguous than mere "center". As shown in Table 3, the average argument length of ACE-E^{++} and ACE-E^{+++} is much longer, but the average trigger length is very similar across the four datasets; it is also evident that single-token arguments make up a large proportion of all arguments, even for ACE-E^{++} and ACE-E^{+++}, so there is a long tail in argument length distribution. For longer arguments, it is more difficult to obtain a perfect match with a gold argument, so we observe decreasing Arg-C scores when EventGraph is evaluated on ACE-E^{++} and ACE-E^{+++}.

To further look into our model's performance on identifying multi-token event arguments, especially those containing more than 5 tokens, we further report Arg-C scores based on token-level span overlap. As shown in Table 5, when we relax argument identification from perfect match to 80% token-level span overlap, the scores of Arg-C increase consistently, especially those of ACE-E^{++} and ACE-E^{+++}, now comparable to the results on ACE-E and ACE-E$^+$.

Dataset	Perfect Match			80% Span Overlap		
	P	R	F1	P	R	F1
ACE05-E	63.4$^{\pm2.7}$	67.3$^{\pm2.0}$	65.3$^{\pm2.2}$	63.9$^{\pm2.4}$	68.5$^{\pm1.7}$	66.2$^{\pm1.9}$
ACE05-E$^+$	64.5$^{\pm1.0}$	66.4$^{\pm2.6}$	65.4$^{\pm1.7}$	65.1$^{\pm0.9}$	67.8$^{\pm2.5}$	66.4$^{\pm1.5}$
ACE05-E^{++}	57.3$^{\pm0.8}$	59.9$^{\pm1.2}$	58.6$^{\pm0.9}$	63.9$^{\pm1.1}$	66.2$^{\pm2.1}$	65.0$^{\pm1.6}$
ACE05-E^{+++}	56.9$^{\pm0.6}$	58.2$^{\pm0.9}$	57.5$^{\pm0.6}$	64.0$^{\pm0.7}$	64.4$^{\pm1.3}$	64.2$^{\pm0.9}$

Table 5: Results of EventGraph on Arg-C, evaluated with perfect match and token-level span overlap.

ACE05-E	ACE05-E$^+$	ACE05-E^{++}	ACE05-E^{+++}
88.8$^{\pm0.4}$	87.9$^{\pm0.5}$	96.2$^{\pm0.2}$	96.5$^{\pm0.6}$

Table 6: Results of EventGraph correctly identifying the presence of event(s) in a sentence.

7 Conclusion

In this paper, we have proposed a new method for event extraction as semantic graph parsing. Our proposed EventGraph has achieved competitive results on ACE2005 for the task of event trigger classification, and obtained new state-of-the-art results for the task of argument role classification. We also provide a graph representation for better visualizing event mentions, and offer an efficient tool to facilitate graph conversion. We create two new datasets from ACE2005, with the full text spans for both triggers and arguments, and offer the corresponding benchmark results. We show that despite adding more and longer text sequences, Event-Graph outperforms previous models tested on more restricted datasets. For future work, we would like to experiment with different pretrained language models, and carry out more detailed error analysis. Our codes and models are released as open-source.

Acknowledgements

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8, Sydney, Australia. Association for Computational Linguistics.

Jeremy Barnes, Robin Kurtz, Stephan Oepen, Lilja Øvrelid, and Erik Velldal. 2021. Structured sentiment analysis as dependency graph parsing. In *Proceedings of the 59th Annual Meeting of the Association for*

11

Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 3387–3402, Online. Association for Computational Linguistics.

Jeremy Barnes, Laura Ana Maria Oberländer, Enrica Troiano, Andrey Kutuzov, Jan Buchmann, Rodrigo Agerri, Lilja Øvrelid, and Erik Velldal. 2022. Semeval-2022 task 10: Structured sentiment analysis. In *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022), Seattle. Association for Computational Linguistics*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

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

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *International Conference on Learning Representations*.

Timothy Dozat and Christopher D. Manning. 2018. Simpler but more accurate semantic dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 484–490, Melbourne, Australia. Association for Computational Linguistics.

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

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online. Association for Computational Linguistics.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2017. A transition-based directed acyclic graph parser for UCCA. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1127–1138, Vancouver, Canada. Association for Computational Linguistics.

I-Hung Hsu, Kuan-Hao Huang, Elizabeth Boschee, Scott Miller, Prem Natarajan, Kai-Wei Chang, and Nanyun Peng. 2022. Degree: A data-efficient generative event extraction model. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of ACL-08: HLT*, pages 254–262, Columbus, Ohio. Association for Computational Linguistics.

Robin Kurtz, Stephan Oepen, and Marco Kuhlmann. 2020. End-to-end negation resolution as graph parsing. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 14–24, Online. Association for Computational Linguistics.

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

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

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

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 (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

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

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

Xiao Liu, Heyan Huang, Ge Shi, and Bo Wang. 2022. Dynamic prefix-tuning for generative template-based event extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5216–5228, Dublin, Ireland. Association for Computational Linguistics.

Xiao Liu, Zhunchen Luo, and Heyan Huang. 2018. Jointly multiple events extraction via attention-based graph information aggregation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1247–1256, Brussels, Belgium. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Yaojie Lu, Hongyu Lin, Jin Xu, Xianpei Han, Jialong Tang, Annan Li, Le Sun, Meng Liao, and Shaoyi Chen. 2021. Text2Event: Controllable sequence-to-structure generation for end-to-end event extraction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2795–2806, Online. Association for Computational Linguistics.

Roser Morante and Walter Daelemans. 2012. ConanDoyle-neg: Annotation of negation cues and their scope in Conan Doyle stories. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 1563–1568, Istanbul, Turkey. European Language Resources Association (ELRA).

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

Stephan Oepen, Omri Abend, Lasha Abzianidze, Johan Bos, Jan Hajic, Daniel Hershcovich, Bin Li, Tim O'Gorman, Nianwen Xue, and Daniel Zeman. 2020. MRP 2020: The second shared task on cross-framework and cross-lingual meaning representation parsing. In *Proceedings of the CoNLL 2020 Shared Task: Cross-Framework Meaning Representation Parsing*, pages 1–22, Online. Association for Computational Linguistics.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinkova, Dan Flickinger, Jan Hajic, and Zdenka Uresova. 2015. SemEval 2015 Task 18: Broad-Coverage Semantic Dependency Parsing. *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 915–926.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Yi Zhang. 2014. SemEval 2014 Task 8: Broad-Coverage Semantic Dependency Parsing. *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 63–72.

Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Jie Ma, Alessandro Achille, RISHITA ANUBHAI, Cicero Nogueira dos Santos, Bing Xiang, and Stefano Soatto. 2021. Structured prediction as translation between augmented natural languages. In *International Conference on Learning Representations*.

David Samuel, Jeremy Barnes, Robin Kurtz, Stephan Oepen, Lilja Øvrelid, and Erik Velldal. 2022. Direct parsing to sentiment graphs. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 470–478, Dublin, Ireland. Association for Computational Linguistics.

David Samuel and Milan Straka. 2020. ÚFAL at MRP 2020: Permutation-invariant semantic parsing in PERIN. In *Proceedings of the CoNLL 2020 Shared Task: Cross-Framework Meaning Representation Parsing*, pages 53–64, Online. Association for Computational Linguistics.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

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)*, pages 5784–5789, Hong Kong, China. 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)*, pages 5777–5783, Hong Kong, China. Association for Computational Linguistics.

Bishan Yang and Tom M. Mitchell. 2016. Joint extraction of events and entities within a document context.

In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 289–299, San Diego, California. Association for Computational Linguistics.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020. Named Entity Recognition as Dependency Parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6470–6476, Online. Association for Computational Linguistics.

A   Training details

We reuse the training settings from the original PERIN system (Samuel and Straka, 2020) whenever possible. The model weights are optimized with AdamW (Loshchilov and Hutter, 2019) following a warmed-up cosine learning rate schedule. We use a pre-trained multi-lingual XLM-R language model implemented by the HuggingFace `transformers` library.[7] The hyperparameter configuration is shown in Table 7, please consult it with our released code for context: https://github.com/huiling-y/EventGraph.

The training was done on a single Nvidia RTX3090 GPU, the runtimes and model sizes (including the fine-tuned language model backbone) for each dataset are given in Table 8.

Hyperparameter	EventGraph
batch_size	16
beta_2	0.98
decoder_learning_rate	1.0e-4
decoder_weight_decay	1.2e-6
dropout_transformer	0.25
dropout_transformer_attention	0.1
encoder	*"xlm-roberta-large"*
encoder_learning_rate	4.0e-6
encoder_weight_decay	0.1
epochs	180
hidden_size_anchor	256
hidden_size_edge_label	256
hidden_size_edge_presence	256
n_transformer_layers	3
query_length	2
warmup_steps	1 000

Table 7: Hyperparameter setting for our system, all four datasets use the same configuration.

Dataset	Runtime	Model size
ACE05-E	20:39 h	341.3 M
ACE05-E$^+$	21:59 h	341.3 M
ACE05-E^{++}	20:06 h	341.3 M
ACE05-E^{+++}	20:03 h	342.0 M

Table 8: The training times and model sizes (number of trainable weights) of all our experiments.

---

[7]https://huggingface.co/docs/transformers/index

B   Data preprocessing

Data splits   All datasets use the same splits[8] for train/dev/test. Out of the 599 documents, 529 documents are used for training, 30 documents for development, and 40 documents for testing.

ACE-E   We use the preprocessing code[9] of Wadden et al. (2019) to obtain the dataset, and they use an older version (v2.0.18) of Spacy[10] for preprocessing.

ACE05-E$^+$ We use the preprocessing code[11] (v0.4.8) of Lin et al. (2020) to obtain the dataset, and they use NLTK[12] for preprocessing.

ACE05-E^{++} and ACE05-E^{+++}   We use the preprocessing code[13] of Wang et al. (2019) to obtain the two datasets, and they use Stanford CoreNLP[14] for preprocessing.

---

[8]https://github.com/dwadden/dygiepp/tree/master/scripts/data/ace-event/event-split
[9]https://github.com/dwadden/dygiepp
[10]https://spacy.io/
[11]http://blender.cs.illinois.edu/software/oneie/
[12]https://www.nltk.org/
[13]https://github.com/thunlp/HMEAE
[14]https://stanfordnlp.github.io/CoreNLP/