A Shallow Semantic Parsing Framework for Event Argument Extraction

Zhunchen Luo[1], Guobin Sui[2], He Zhao[3], and Xiaosong Li[1](✉)

[1] Information Research Center of Military Science,
PLA Academy of Military Science, Beijing, China
zhunchenluo@gmail.com, 164832732@qq.com

[2] School of Computer Science and Engineering, Beihang University , Beijing, China
suigb@buaa.edu.cn

[3] School of Computer Science, Beijing Institute of Technology, Beijing, China
zhaohe1995@outlook.com

Abstract. Currently, many state-of-the-art event argument extraction systems are still based on an unrealistic assumption that gold-standard entity mentions are provided in advance. One popular solution of jointly extracting entities and events is to detect the entity mentions using sequence labeling approaches. However, this methods may ignore the syntactic relationship among triggers and arguments. We find that the constituents in the parse tree structure may help capture the internal relationship between words in an event argument. Besides, the predicate and the corresponding predicate arguments, which are mostly ignored in existing approaches, may provide more potential to represent the close relationship. In this paper, we address the event argument extraction problem in a more actual scene where the entity information is unavailable. Moreover, instead of using word-level sequence labeling approaches, we propose a shallow semantic parsing framework to extract event arguments with the event trigger as the predicate and the event arguments as the predicate arguments. In specific, we design and compare different features for the proposed model. The experimental results show that our approach advances state-of-the-arts with remarkable gains and achieves the best F1 score on the ACE 2005 dataset.

Keywords: Event argument extraction · Entity · Shallow semantic parsing

1 Introduction

In recent years, event extraction, which aims to discover event triggers with specific types and their arguments, has attracted increasing interests in the research communities. In the Automatic Context Extraction (ACE) event extraction program[1], an event is represented as a structure, comprising an event trigger and a set of arguments. This work tackles the event argument extraction (EAE) task, which is a crucial part of event extraction (EE) and focuses on identifying event arguments and categorizing them into

[1] https://www.ldc.upenn.edu/collaborations/past-projects/ace.

C. Douligeris et al. (Eds.): KSEM 2019, LNAI 11776, pp. 88–96, 2019.
https://doi.org/10.1007/978-3-030-29563-9_9

roles in the events triggered by the given triggers. The following is an example to illustrate the EAE task.

E1: *Controversial PLO leader Yasser Arafat* ***died*** *in a Paris hospital on Sunday*.
In this example, given the sentence **E1** and the trigger word "***died***" which triggered a *Die* event, an EAE system is expected to detect "*Controversial PLO leader Yasser Arafat*" as a *Victim* argument, "*a Paris hospital*" as a *Place* argument, and "*Sunday*" as a *Time* argument.

Current state-of-the-art methods [1–7] often model the EAE task as an entity-level classification problem with an unrealistic assumption that the gold-standard entity annotations are provided in advance. However, in real-life applications, the entity-level classification approach can hardly be directly applied to EAE, because the gold-standard entity annotations are not always available. Therefore, how to recognize the entity mentions remains a challenging issue. Besides, previous research on joint entity and event extraction models the entity mention recognization problem as a word-level sequence labeling problem by classifying the words using the BIO schema and then feed the sequence labeling results into the later EAE models [8]. However, in those entity-event pipeline frameworks, the errors in entity mention recognition will propagate and affect the final extraction result. What's more, there are some shortages of the current sequence labeling approaches to EAE. First, the words in an argument are assumed to be independent to each other. So that the internal mutual relationship between words are ignored for extracting a complete event argument. For example, in **E1**, the *Victim* argument is "*Controversial PLO leader Yasser Arafat*", and it contains several words. If ignoring the relationship among words, the modified parts "*Controversial PLO*" might be ignored, hence an incomplete argument "*leader Yasser Arafat*" will be extracted.

E2: *Melony Marshall was* ***married*** *just a month before she* ***left*** *for Iraq*.
Second, the sequence labeling approaches often use the contextual information of the arguments but ignore the relationship among the arguments and the event triggers. For example, in **E2**, there are two events which are triggered by "***married***" and "***left***", respectively. It is essential to know the relationship between the phrase "*just a month before*" and each trigger to determine which event trigger it belongs to. Therefore, it remains immature to model the close relationship between an event trigger and its arguments in current approaches.

In this paper, we propose a novel approach which simultaneously extracts event arguments and classifies them into the roles in the events to solve the above two main critical issues. We model this task from a parse tree structure perspective and adopt a shallow semantic parsing and classifying framework. More specifically, first we use the abundant constituents provided by the pars tree structure to capture the internal relationship between words in an event argument. Second, we use a rule-based argument merger to find the appropriate constituents which indicates the event argument candidates in the given parse tree. And then, we introduce an argument pruning procedure to filter the candidates which are most likely non-arguments of a predicate. Finally, we use the well-designed parse tree structure features to infer whether the argument is related to the event and the corresponding role. In experiments, we systematically conduct comparisons on a widely used benchmark dataset, ACE2005[2].

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

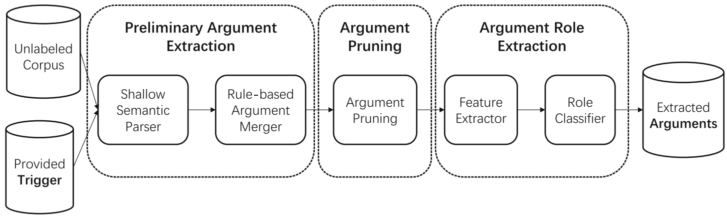

Fig. 1. An overview of the shallow semantic parsing and classifying framework.

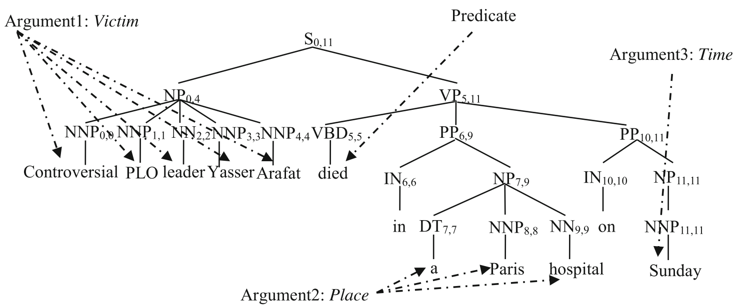

Fig. 2. Illustration of an event trigger (predicate) and its corresponding event arguments in a parse tree of the sentence **E1** (*Controversial PLO leader Yasser Arafat* ***died*** *in a Paris hospital on Sunday*).

The experimental results demonstrate that our proposed approach is effective for the EAE task, and it outperforms state-of-the-art approaches with remarkable gains.

2 Approach

We formulate the EAE task as a shallow semantic parsing problem and solve it in a simplified shallow semantic parsing and classifying framework. As shown in Fig. 1, our proposed framework can be divided into three subsections: Preliminary Argument Extraction, Argument Pruning, and Argument Role Extraction, which will be introduced respectively.

Preliminary Argument Extraction

To capture the internal and mutual syntactic relationship among triggers and arguments, we convert the original word sequences to semantic parse trees given the event triggers as predicates. Preliminary event arguments are recast as the predicate arguments in parse trees. The task of shallow semantic parsing is to map all the constituents in the sentence into their corresponding semantic arguments (roles) of the predicate given the sentence's parse tree and a predicate in it. For EAE, the event trigger is considered as the predicate, while the event arguments are mapped into the predicate's arguments. Figure 2 shows the parsing tree structure of the sentence in **E1**, together with the event trigger and its corresponding event arguments. To find the proper constituents in parse trees containing those separated tokens respectively, given the trigger and the sentence

Table 1. Basic Features with Regard to Fig. 2. And the contents between the parentheses are corresponding extracted basic features with regard to the sentence **E1**.

Category	Features	Description
B-Pre	**B-Pre1**	The predicate and, its POS and type. (died, VBD, Life/Die)
B-Arg	**B-Arg1**	The argument candidate and its POS. (controversial plo leader Yasser Arafat, NNP_NNP_NN_NNP_NNP)
	B-Arg2	The headword of the argument candidate and its POS. (controversial, NNP)
	B-Arg3	The last word of the argument candidate and its POS. (Arafat, NNP)
	B-Arg4	The left word of the argument candidate and its POS. (null, null)
	B-Arg5	The right word of the argument candidate and its POS. (died, VBD)
	B-Arg6	Whether the argument candidate includes a time word. (no)
B-Arg-Pre	**B-Arg-Pre1**	The predicate and the positional relationship of the argument candidate with the predicate ("left" or "right"). (died_left)
	B-Arg-Pre2	The distance from the headword of the argument candidate to the predicate. (5)

which contains m words: $word_1$, ..., $word_m$, we propose a rule-based argument merger consisting of the following three heuristic rules to merge the separated tokens to the appropriate constituents indicating event arguments in the given parse tree.

PAE-Rule(1): The trigger itself and all of its ancestral components are non-arguments.

PAE-Rule(2): If all child constituents of constituent X are recognized as arguments, X is labeled as an argument, and all its child constituents are re-labeled as non-arguments.

PAE-Rule(3): If not all of the child constituents of constituent X are recognized as arguments, X is labeled as a non-argument.

After applying the rule-based argument merger, we can obtain event argument candidates given a particular event trigger.

Argument Pruning

However, the previous preliminary argument extraction step may bring in extra noises of event argument candidates which are most likely non-arguments of a predicate. To filter the noises, we introduce an argument pruning procedure based on three heuristic rules to obtain the extracted event argument candidates. The heuristics in the argument pruning are executed sequentially in the given order:

AP-Rule(1): The constituents which contain more than five words are filtered out as non-arguments and instead, their child constituents are considered as argument candidates.

AP-Rule(2): The predicate constituent itself and its ancestral constituents in the parse tree are filtered out as non-arguments.

AP-Rule(3): The constituents which have no brother constituents are filtered out as non-arguments.

Table 2. Syntactic Features with Regard to Fig. 2. And the contents between the parentheses are corresponding extracted syntactic features with regard to the sentence **E1**.

Category	Features	Description
S-Arg	**S-Arg1**	The node name of the argument candidate. (NP)
	S-Arg2	Whether the node of the argument candidate is the largest NP node including a time word. (no)
S-Arg-Pre	**S-Arg-Pre1**	The path from the headword of the argument candidate to the predicate. (NP<S>VP>VBD)
	S-Arg-Pre2	The number of the nodes in S-Arg-Pre1. (4)
	S-Arg-Pre3	The partial syntactic path from the argument candidate node to the least governing node of both the argument candidate and the predicate. (NP<S)
	S-Arg-Pre4	The partial syntactic path from the predicate to the least governing node of both the argument candidate and the predicate. (VBD<VP<S)
	S-Arg-Pre5	Whether there is a clause tag (S) between the predicate and the argument candidate nodes. (yes)
	S-Arg-Pre6	Whether there are more than 3 clause tags (S) between the predicate and the argument candidate nodes. (no)

Argument Role Extraction

Event argument extraction (EAE) involves typically two sub-tasks, namely argument identification, and argument classification. The former is to identify whether an event argument has been playing a particular role in the event triggered by a given trigger. The latter, however, is to categorize an event argument to the role played in that event. In our approach, we deal with these two sub-tasks together by considering the non-argument (the event argument has not played a specific role in that event) as an extra and particular category of event argument roles. Thus, the whole argument extraction is formulated as an argument-level multi-category classification problem.

To address the multi-category classification problem, we use the conditional random fields (CRFs) and exploit two groups of features: **basic features** and **syntactic features**, as shown in Table 1. The features are categorized into three groups: the predicate-related features, B-Pre; the argument-related features, B-Arg; the features related to both the argument and predicate, B-Arg-Pre. Besides the basic features, various kinds of syntactic features in Table 2 are explored to capture more details regarding the argument candidate and the predicate. In the same spirit, we categorize the syntactic features into two groups, namely S-Arg and S-Arg-Pre. The first group contains the argument-related features and the second group contains the argument and predicate related features.

3 Experiments

In our approach, argument pruning is an important strategy to reduce the amount of argument candidates. Table 3 shows the numbers of argument candidates when the pruning rules as discussed in Sect. 3.2 are used. From this table, we can see that the first

Table 3. The effectiveness of the argument pruning rules

Action	#Argument candidates (Percentage)
Without Pruning	20920 (100%)
After **AP-Rule(1)**	16719 (79.9%)
After **AP-Rule(1)+(2)**	15841 (75.7%)
After **AP-Rule(1)+(2)+(3)**	14192 (67.8%)

Table 4. Performance comparison between word-sequence and parsing level approaches to argument extraction

Features	Argument identification			Argument classification		
	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)
Word-sequence level (Basic)	63.36	38.52	47.91	60.14	**36.55**	45.47
Parsing level (Basic)	66.82	40.06	50.09	**82.17**	32.91	47.00
Parsing level (Basic + Syntactic)	**67.48**	**42.44**	**52.11**	81.52	34.59	**48.57**

rule **AP-Rule(1)** (filtering the constituent with more than five words) is most effective, which filters about 20% candidates. When all rules are used, 32.2% argument candidates are filtered.

Experimental Settings

For dataset, we use the English corpus for event extraction in ACE2005, which involves 8 types and 33 subtypes of events. Among all the 599 documents, we use 559 of them for training and the rest 40 for test. Different from previous studies [1–7], the gold-standard entity information is not available in our experiment.

We use the Stanford Parser[3] to get the POS tags and shallow semantic parse trees of the sentences. And for the multi-category classification problem, we use the conditional random fields (CRFs), which is implemented with a public tool CRF++[4].

For evaluation, we use the following criteria to determine the correctness of an predicted event mention: (1) An argument is correctly identified if its event subtype and offsets match any of the reference argument mentions. (2) An argument is correctly identified and classified if its event subtype, offsets and argument role match any of the reference argument mentions. Finally we use Precision (P), Recall (R) and F_1-measure (F_1) to evaluate the performance of event argument extraction.

Experimental Results

Table 4 shows the comparison between word-sequence and parsing level approaches to argument extraction. We can see that even using the basic features only, our parsing level approach is apparently superior to the word-sequence level approach in both argument identification, 47.91% vs. 50.09% and argument classification, 45.47% vs. 47.00%. Significance test with t-test shows that the improvement of our parsing level

[3] https://nlp.stanford.edu/software/lex-parser.html#Citing.
[4] https://code.google.com/p/crfpp/.

Table 5. Contribution of syntactic features in argument identification

Feature	P(%)	R(%)	F1(%)	Feature	P(%)	R(%)	F1(%)
Basic	66.82	40.06	50.09	Basic	66.82	40.06	50.09
+S-Arg1	66.36	40.34	50.17	+S-Arg1	66.36	40.34	50.17
+S-Arg2	66.51	39.22	49.34	+S-Arg2	66.51	39.22	49.34
+S-Arg-Pre1	**68.35**	39.92	50.40	+S-Arg-Pre1	**68.35**	39.92	50.40
+S-Arg-Pre2	65.32	40.62	50.09	+S-Arg-Pre2	65.32	40.62	50.09
+S-Arg-Pre3	67.12	41.74	51.47	+S-Arg-Pre3	67.12	41.74	51.47
+S-Arg-Pre4	67.50	41.60	51.47	+S-Arg-Pre4	67.50	41.60	51.47
+S-Arg-Pre5	66.67	39.78	49.82	+S-Arg-Pre4	67.50	41.60	51.47
+S-Arg-Pre6	67.61	40.34	50.53	+S-Arg-Pre6	67.61	40.34	50.53
All	67.48	**42.44**	**52.11**	All	67.48	**42.44**	**52.11**

Table 6. Comparison to the state-of-the-art

Approach	Performance on argument classification
Li et al. [5] with gold-standard entity information	52.7%
Li et al. [5] with entity extraction system	41.8%
Nguyen et al. [7] with gold-standard entity information	55.4%
Nguyen et al. [7] with entity extraction system	45.3%
Yang et al. [8]	48.4%
Our Approach	**48.6%**

approach (Basic+Syntatic) over the word-level approach is significant (p-value¡0.05). Especially, in argument classification, our parsing level approach outperforms word-sequence level approach in precision with a very large margin, i.e., 22.03%. Once the syntactic features are leveraged, the performance of our approach could be further improved.

Table 5 (left) shows the contribution of syntactic features in argument identification. The syntactic path features of S-Arg-Pre3, S-Arg-Pre4, and S-Arg-Pre6 are very effective for argument identification. Compared to using basic features only, using all these syntactic features yields a better performance in both precision and recall. And Table 5 (right) shows the contribution of syntactic features in argument classification. We can see that using all syntactic features is superior in terms of recall but a bit inferior in terms of precision. Overall, using syntactic features improves the performance in F-measure.

Table 6 shows the result comparison to the results provided by the state-of-the-art work of Li et al. [5], Nguyen et al. [7], and Yang et al. [8]. Our approach performs 3%-6% better than using an automatic system for extracting entities. Besides, our approach also outperforms the method of jointly extracting entities, event triggers, and event arguments. It can be seen the internal and mutual syntactic relationship among triggers and arguments is essential. Although our approach performs worse than their approach when gold-standard entity information is used, our approach does not employ the global features as their approach applies.

4 Conclusion

In this paper, we focus on a more realistic scenario in event extraction when the gold-standard entity annotations are not available in the test phase. We present a novel approach to event argument extraction by formulating it as a shallow semantic parsing problem. And various kinds of basic and syntactic features are employed to extract event arguments. Empirical studies demonstrate that our approach considerably outperforms traditional sequence labeling approaches. In the future work, we will improve our approach by leveraging a stronger parser or introducing some operations to recall more event arguments. And we will exploit higher-level knowledge, such as the cross-event and the cross-entity information.

Acknowledgements. This work was supported by the National Natural Science Foundation of China (No. 61602490).

References

1. Chen, Y., Xu, L., Liu, K., Zeng, D., Zhao, J.: Event extraction via dynamic multi-pooling convolutional neural networks. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, Volume 1: Long Papers, Beijing, China, 26–31 July 2015, pp. 167–176 (2015). http://aclweb.org/anthology/P/P15/P15-1017.pdf
2. Hong, Y., Zhang, J., Ma, B., Yao, J., Zhou, G., Zhu, Q.: Using cross-entity inference to improve event extraction. In: The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, Portland, Oregon, USA, 19–24 June 2011, pp. 1127–1136 (2011). http://www.aclweb.org/anthology/P11-1113
3. Ji, H., Grishman, R.: Refining event extraction through cross-document inference. In: ACL 2008, Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, Columbus, Ohio, USA, 15–20 June 2008, pp. 254–262 (2008). http://www.aclweb.org/anthology/P08-1030
4. Li, Q., Ji, H., Hong, Y., Li, S.: Constructing information networks using one single model. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL, 25–29 October 2014, pp. 1846–1851 (2014). http://aclweb.org/anthology/D/D14/D14-1198.pdf
5. Li, Q., Ji, H., Huang, L.: Joint event extraction via structured prediction with global features. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, Volume 1: Long Papers, Sofia, Bulgaria, 4–9 August 2013, pp. 73–82 (2013). http://aclweb.org/anthology/P/P13/P13-1008.pdf

6. Liao, S., Grishman, R.: Using document level cross-event inference to improve event extraction. In: ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, Uppsala, Sweden, 11–16 July 2010, pp. 789–797 (2010). http://www.aclweb.org/anthology/P10-1081
7. Nguyen, T.H., Cho, K., Grishman, R.: Joint event extraction via recurrent neural networks. In: NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, 12–17 June 2016, pp. 300–309 (2016). http://aclweb.org/anthology/N/N16/N16-1034.pdf
8. Yang, B., Mitchell, T.M.: Joint extraction of events and entities within a document context. In: NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, 12–17 June 2016, pp. 289–299 (2016). http://aclweb.org/anthology/N/N16/N16-1033.pdf