An End-to-End Event Extraction Framework with Aligning Mechanism on Few Labeled Data

Qian Li[1] , **Jianxin Li**[2*] , **Hao Peng**[3] , **Yuanxing Ning**[4] , **Jiachun Li**[5] and **Lihong Wang**[6]

[1]Beihang University

[2]National Computer Network Emergency Response Technical Team Coordination Center of China

{liqian, lijx}@act.buaa.edu.cn, penghao@act.buaa.edu.cn, wlh@isc.org.cn

Abstract

There is massive amount of news on events every day. Due to the high cost of labeled data, it is very important to study how to detecting event and extracting its arguments from a small amount of labeled data and a large amount of unlabeled data. In this paper, we present an end-to-end framework for detecting, classifying event and extracting events arguments. This model exploits a multi-task learning approach, in which a pre-trained BERT model and a graph attention network is used to encode context and structure information of the sentences, and the encoded information are shared by event detection, type classification and detection tasks. The question generator generates a question set of all arguments for each event type, based on the categorized event type. For event arguments extracting, we use the same graph attention network and a pre-trained BERT model for encode the sentences and the question of each argument. To increase the effective labeled data, the high confidence results of event classification and argument recognition model are added to the input data. Our experiments show that our approach outperforms other methods.

1 Introduction

Event extraction is an important subtask in the field of information extraction, which aims to extract structured events information from text. The problem of event extraction requires detecting the event trigger and extracting its corresponding arguments. For instance, given the following sentence:

Sample: Baghdad, a cameraman died when an American tank fired on the Palestine Hotel.

Event extraction needs to recognize the two events (*Die* and *Attack*), triggered by the words "died" and "fired" respectively, as shown in Figure 1. For *Die* event type, we recognize that "Baghdad", "cameraman" and "American tank" take on the event argument roles *Place*, *Victim* and *Instrument* respectively . For *Attack*, "Baghdad" and "American tank" take on the event argument roles *Place* and *Instrument* respectively. And "cameraman" and "Palestine Hotel" take on the event argument roles *Target*.

The goal of event extraction is to firstly classify the event type of each text. Through structured event framework according to the corresponding event type, event extraction then automatically extracts each argument of the event that is originally scattered in the content from the text, and gives the relationship among the arguments. We perform event extraction using two-stage process:

- Event classification: Event classification is to determine whether each sentence is an event. Furthermore, if the sentence is an event, which one or several event types the sentence belongs to. Therefore, it is a multi-label text classification task to classify the type of each event.
- Trigger identification and argument extraction: It is generally considered that the trigger is the core unit in event extraction that can clearly express the occurrence of an event. However, not all events are the most important triggers. Here, we do not distinguish between event triggers and event arguments. We consider triggers and event arguments to be equally important.

It takes a lot of labor and time to construct sufficient event extraction datasets. The construction of event extraction dataset needs to design different templates for different event types. The same sentence may belong to multiple event types, and event arguments may be missing in the sentence, which brings difficulties to the annotation data, Furthermore, it also requires lots of manual work to evaluate the annotation performance of data set. The currently widely recognized dataset is Automatic Content Extraction (ACE) [Doddington et al.,], which contains 599 annotated texts with a total of 33 event types, such as Attack and Justice events. The existing event extraction dataset with event templates have only few labeled data. Huang et al. [Huang et al.,] design a transferable architecture of structural and compositional neural networks to jointly represent and map event mentions and types into a shared semantic space. This paper adds dataset by designing a problem generator and constructing more problems. Simultaneously, we design a feedback mechanism for the prediction results of all arguments, and the results with high confidence in the prediction are added to the training data as pseudo label.

However, the event extraction task is complex and multi-

*Contact Author

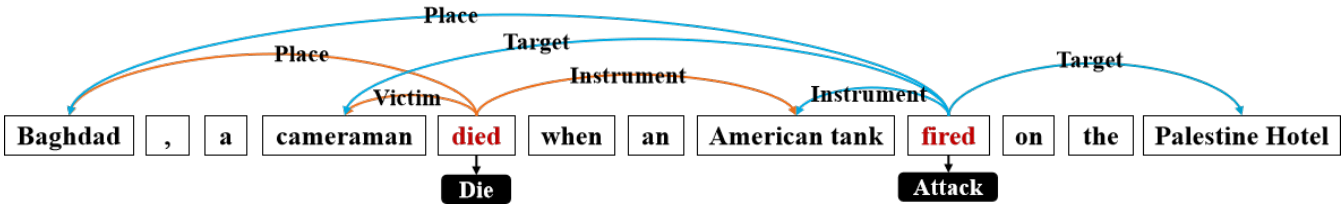

Figure 1: An diagram of event extraction. The example can be divided into two type of event. The type of Die is triggered by 'died' with three argument roles of 'Place', 'Victim' and 'Instrument' and the type of Attack is triggered by 'fired' with three argument roles of 'Place', 'Target' and 'Instrument'.

task. It requires identifying the type of event, finding the event trigger and event argument, and classifying the role of each event argument. Multi-task learning has been applied in many NLP tasks, and has shown its ability to improve the performance of these tasks. This paper presents a joint learning model for detecting, classifying and extracting events. We fine-tune a pre-trained BERT model [Devlin et al.,] to generate text representation, which is shared by all the three tasks. This shared model will extract the common information and patterns among these three tasks, and the pre-trained BERT model lets us exploit the grammar and semantic information of tokens learned.

Furthermore, event extraction needs to solve the overlapping and dispersions of arguments, which requires the model to accurately learn the semantic relationship of text and the multi-semantic representation of words. The machine reading comprehension based event extraction identifies arguments in turn by constructing problems. Due to the influence of different event types and recognized arguments, the model learns multi-semantic representation of words and avoids the influence of overlapping arguments on event extraction performance. The machine reading comprehension event extraction can locate the answer according to the question and retrieve in the full text, which can overcome the difficulty of distance dependence in the event extraction task. This paper implements the event argument extraction task using a machine reading comprehension model.

Different from existing type event extraction method of reading comprehension, our question generator constructs the question based on the event type and the predicted event argument with high confidence to identify the next argument. In this way, we can make full use of the correlation between the arguments. If the confidence of the prediction results is low, we will make another prediction based on the existing argument results after the identification of other arguments is completed.

In this paper, we simultaneously address the few labeled data problem as well as the argument overlap problem in the event extraction. To be specific, we contribute to

- designing a question generator and a feedback mechanism for the prediction results of all arguments and the results with high confidence in the prediction are added to the training data as pseudo label.
- presenting a multi-task model for event type classification, event detection and argument extraction task with a shared pre-trained BERT model to extract the common information and patterns among these three tasks and exploit the grammar and semantic information of tokens learned.
- designing a machine reading comprehension model that can make full use of the correlation between arguments through constructing the question based on the event type and the predicted event argument with high confidence to identify the next argument.
- We further conduct substantial experiments and verify that the proposed method reaches the excellent performance by using sequence question generator and aligning mechanism.

2 Related Work

Pipeline event extraction In the past, the pipeline modeling method was used to model the association between events. The pipeline idea, which treats all sub-tasks as independent classification problems, is called the pipeline method. To automatically extract lexical and sentence-level features without using complex natural language processing tools, Chen et al. [Chen et al.,] introduce a word representation model, called DMCNN, to capture the meaningful semantic rules of words and adopt a framework based on a convolutional neural network (CNN) to capture sentence-level clues. Nguyen et al. [Nguyen and Grishman, a] use CNN to investigate the event detection task, which overcomes complex feature engineering and error propagation limitations compared with traditional feature-based approaches. But it relies extensively on other supervised modules and manual resources to obtain features. It is significantly superior to the feature-based method in terms of cross-domain generalization performance. Furthermore, to consider non-consecutive k-grams, Nguyen et al. [Nguyen and Grishman, b] introduce non-consecutive CNN. The application of tree structure and sequence structure in a neural network has better performance than a sequential structure.

Pre-training event extraction. Pre-training has been widely used in natural language processing (NLP) tasks to learn better language representation, and several new pre-trained models have been published recently, such as BERT [Devlin et al.,], XLNet [Yang et al.,], RoBERTa [Liu et al.,], ALBERT [Lan et al.,], ELMO [Peters et al.,], etc. The pre-training on large amount of unlabeled data and fine-tuning with small scale labeled data are helpful for many tasks, and it is also used in the encoder part of our model in this

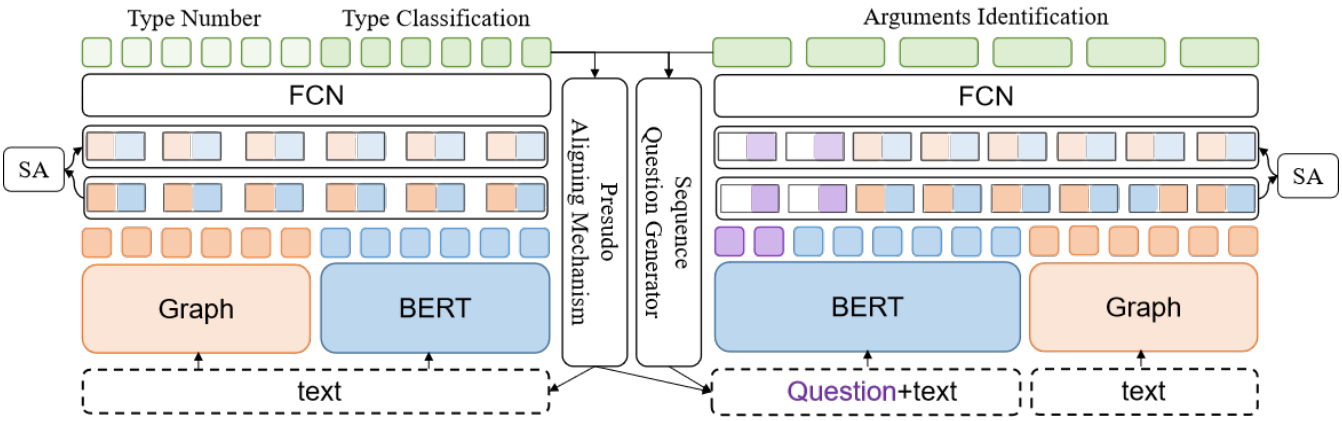

Figure 2: The the architecture of the Event Extraction Framework.

work. However, it is challenging to exploit one argument that plays different roles in various events to do better in event extraction. Yang et al. [Yang et al., 2019] propose an event extraction model to overcome the roles overlap problem by separating the argument prediction in terms of roles. Moreover, to address insufficient training data, they propose a method to automatically generate labeled data by editing prototypes and screen out generated samples by ranking the quality. They present a framework, Pre-trained Language Model-based Event Extractor (PLMEE) [Yang et al., 2019], to promote event extraction by using a combination of an extraction model and a generation method based on pre-trained language models.

3 The Unified Model

Figure 2 shows the high-level model structure. The model consists of three parts: event classification, sequence problem generation and argument extraction. Event classification detects whether the input text is an event and classifies the event type to which the text belongs. The sequential problem generator automatically generates the problem based on the event type and the predicted high confidence argument results. Argument extraction takes text and problem as input predictors, and then the predictors with high confidence are added to the training set through label alignment mechanism.

3.1 Event Classification

The event classification model determines whether the current text contains events. If the text does not contain an event, NULL is output and the subsequent modules are not executed. Otherwise, detect what kinds of events the text contains.

In this work, the model input text is passed into a Graph Attention Networks (GAT) [Velickovic et al.,] model and a BERT [Devlin et al.,] model respectively, and the structured knowledge and context knowledge of the text are learned respectively. In order to improve event classification performance, we construct an event category quantity prediction task. The model calculates the combined loss of the two tasks of event classification and event category number prediction, which enables the model to take into account the difference between the current prediction error due to the prediction error of event category or the prediction error of event category is less or more.

A. Lexicon-Based Graph Neural Networks on Texts

We make a graph of each text, each word is treated as a node in the graph. We use lexical knowledge to concatenate characters and propose a global relay node to capture local composition and potential word boundaries that we refer to the method of constructing edges in [Gui et al.,]. There are four kinds of connecting edges between nodes, as shown in Figure 3.

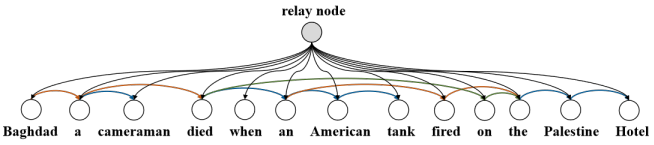

Figure 3: The the architecture of the Event Extraction GAT. The black side is the connection of the relay node and all words, the orange side is the connection of words, the blue side is the connection of words, and the green side is the connection of words with high co-occurrence probability.

A schematic diagram of constructing a line of a sentence is given, in which the gray node represents the relay node and each other node represents a word in the sentence. The first kind of connection we construct is the word-in-word connection, that is, the words in a phrase are connected sequentially until they are connected to the last word, such as the edge between 'a' and 'cameraman'. The second way is to create a line between words, like the orange edges in Figure 3, such as the edge between 'a' and 'died'. The concrete connection mode is that the first word of the former word is connected with the first word of the latter word, and each edge represents the potential characteristics of the word that may exist. We also designed a relay node, which is connected to all the edges and nodes in the graph, and used to gather the information of all the edges and points, thus eliminating the boundary ambiguity between words. The global relay node is connected to all words, with the black edge in Figure 3, which enables the relay node to learn global information

about the document. It is used to gather information of all edges and points, thus it eliminates boundary ambiguity between words. Therefore, the representation of the relay node can be regarded as the representation of the document. Finally, we calculate the co-occurrence probability of words in the data set, and construct a continuous edge between nodes with high contribution probability, such as the edge between 'died' and 'on' in Figure 3.

Then, according to the constructed graph, a GAT model is trained to learn the structural representation of each node in the text.

According to the constructed graph, train a GAT model and learn the structural representation of each node in the text. We assume that there is a text with N words, and the GAT model has an M-level graph attention layer, then the input node represents as

$$\mathbf{IG} = \left\{\vec{ig}_1, \vec{ig}_2, \ldots, \vec{ig}_N\right\}, \vec{ig}_i \in R^F$$

.

The representation set of output nodes of GAT model is

$$\mathbf{OG} = \left\{\vec{ig}_1', \vec{ig}_2', \ldots, \vec{ig}_N'\right\}, \vec{ig}_i' \in R^{F'}$$

.

B. BERT on Texts

The text is also passed into a BERT model, which first learns the context representation of each word. Then, we splicing the word representation output by GAT model and BERT model to get a new representation of each word.

$$P_s^r(t) = \text{Softmax}\left(W_s^r \cdot \mathcal{B}(t)\right)$$

$$P = FCN(loss_{GAT} + loss_{BERT})$$

We add a self-attention mechanism to learn new contextual representations of words.

$$P = SA(loss_{GAT} + loss_{BERT})$$

$$\mathcal{L}_s = \sum_{r \in \mathcal{R}} \frac{I(r,v)}{|\mathcal{S}|} \text{CE}\left(P_s^r, \boldsymbol{y}_s^r\right)$$

$$\mathcal{L}_e \& = \sum_{r \in \mathcal{R}} \frac{I(r,v)}{|\mathcal{S}|} \text{CE}\left(P_e^r, \boldsymbol{y}_e^r\right)$$

Finally, after passing through the FCN layer, the model outputs whether the current text contains an event, and if so, the event type.

$$P_e^r(t) = \text{Softmax}\left(W_e^r \cdot \mathcal{B}(t)\right)$$

$$P = FCN(loss_{GAT} + loss_{BERT})$$

C. Multi-task Joint Loss

The text is also passed into a BERT model, which first learns the context representation of each word. Then, we splicing the word representation output by GAT model and BERT model to get a new representation of each word.

3.2 Sequence Question Generation

According to the event type to which the text predicted by the event classification model belongs and the event template corresponding to the event type, we generate the problem set of each argument role in turn. Due to the limited tag data in the event extraction data set, we construct multiple problems for each argument.

$$Question = FCN(loss_{GAT} + loss_{BERT})$$

Algorithm 1 QuestionGeneration(type, roles):

In: type, roles.
Out: questions
Initiate: $max \leftarrow roles[0]$.
$question[max] \leftarrow Q(type, template)$
$Pmax \leftarrow ExtractionP(question[max], text)$
for $role$ **in** $roles$ **do**
    $question[role] \leftarrow Q(type, template)$
    $P(role) \leftarrow ExtractionP(question[role], text)$
    if $P(role) > Pmax$ and $P(role) > P$ **then**
        $Rmax \leftarrow role$
        $Pmax \leftarrow P(role)$
    end if
end for
if $Pmax > P$ **then**
    $question[Rmax] \leftarrow Q(type, template)$
    $Label \leftarrow Extraction(question[Pmax], text)$
end if
roles.pop(Rmax)
QuestionGeneration(type,roles)

First, we identify the first argument, and the construction of the problem contains only the event type information and the argument role to be identified. We identify all the event arguments in turn and select the one with the highest confidence as the first event argument.

$$Selection = FCN(loss_{GAT} + loss_{BERT})$$

$$MRC = FCN(loss_{GAT} + loss_{BERT})$$

$$\mathcal{E}(a) = \frac{1}{|a|} \sum_{t \in a} \mathcal{E}(t)$$

Then, we construct the problem identified by the second and second arguments, which consists of the event type and the first argument, and then identify the next argument with the highest confidence. And so on until all arguments are identified. We set a confidence threshold, which increases arguments only when reached. If no argument is added in one round of argument recognition, that is, the remaining argument recognition results have low confidence, we add all the prediction results at the same time.

3.3 Sequence Argument Extraction

The input of the argument extraction model consists of two parts: the problem generated by the sequence problem generation model and the input text. Since the event extraction data set has only a small amount of labeled data, in order to

Table 1: Event classification, argument identification and argument role classification result on the ACE2005 test set. P(recision), R(ecall), and F1 obtained by models on the ACE 2005 dataset. Best results are bolded. Using the full training set improves F1 performance over using the partial training set on all metrics except argument identification (equal). Pre-training on the recast SQuAD 2.0 dataset improves F1 performance over no pre-training on all metrics.

	Event Classification			Argument Identification			Argument Role Classification		
	P	R	F1	P	R	F1	P	R	F1
JMEE [Liu et al., 2018]	76.3	71.4	73.7	-	-	-	66.8	54.9	60.3
Joint3EE [Nguyen and Nguyen, 2019]	68.00	71.80	69.80	-	-	-	52.10	52.10	52.10
GAIL-ELMo [Zhang et al., 2019]	74.8	69.4	72.0	-	-	-	61.6	45.7	52.4
PLMEE [Yang et al., 2019]	81.0	80.4	80.7	71.4	60.1	65.3	62.3	54.2	58.0
Chen et al. [Chen et al., 2019]	66.7	74.7	70.5	44.9	41.2	43.0	44.3	40.7	42.4
Du et al. [Du and Cardie, 2020]	71.12	73.7	72.39	58.9	52.08	55.29	56.77	50.24	53.31
MQAEE [Li et al., 2020]	-	-	73.8	-	-	56.7	-	-	55.0
Our model	**81.2**	74.62	78.56	-	-	-	61.48	54.43	**58.32**

Algorithm 2 LabelGeneration(type, roles):

In: type, roles.
Out: Labels
Initiate: $max \leftarrow roles[0]$.
$question[max] \leftarrow Q(type, template)$
$Pmax \leftarrow ExtractionP(question[max], text)$
for $role$ **in** $roles$ **do**
    $question[role] \leftarrow Q(type, template)$
    $P(role) \leftarrow ExtractionP(question[role], text)$
    if $P(role) > Pmax$ and $P(role) > P$ **then**
        $Rmax \leftarrow role$
        $Pmax \leftarrow P(role)$
    end if
end for
if $Pmax > P$ **then**
    $question[Rmax] \leftarrow Q(type, template)$
    $Label \leftarrow Extraction(question[Pmax], text)$
end if
roles.pop(Rmax)
LabelGeneration(type,roles)

make full use of the knowledge of existing data, we share the event classification and representation of the Chinese version of the argument extraction model.

$$MRC = FCN(loss_{GAT} + loss_{BERT})$$

According to the results of the event classification, the template corresponding to the event type (that is, the argument roles that the event type contains) is determined. For each argument role, the sequential problem generation module generates the problem set. In the argument extraction model, we design a machine reading comprehension model, inputting a question and corresponding text one at a time.

$$MRC = FCN(loss_{GAT} + loss_{BERT})$$

$$MRC = FCN(loss_{GAT} + loss_{BERT})$$

The problem is fed into a BERT model. Text was input into a GAT model to learn the structured representation of text and the BERT model to learn the context representation of text respectively, and then the results of the two models were splited together through a self-attention mechanism. Finally, the starting and ending positions of the argument are predicted by the FCN layer.

$$Label = FCN(loss_{GAT} + loss_{BERT})$$

We have added a tag alignment mechanism. We believe that the arguments with high confidence predicted in the argument extraction model are reliable, and these results are very close to the real results. In the case of insufficient label data in the event extraction data set, the result with high confidence is taken as a label and added to the training data.

$$MRC = FCN(loss_{GAT} + loss_{BERT})$$

$$MRC = FCN(loss_{GAT} + loss_{BERT})$$

Since it is impossible to learn the correlation between arguments by extracting all arguments simultaneously, we design a sequence argument extraction model. We predict the arguments in turn, add the high confidence in the prediction results to the problem, and provide additional information for extracting the next argument. For an argument role with low confidence, we identify all arguments in the event template after one round, and then re-identify those arguments with low confidence, so that the argument information identified in the previous round can be used.

$$\mathcal{L}_s = \frac{1}{|\mathcal{R}| \times |\mathcal{S}|} \sum_{r \in \mathcal{R}} \mathrm{CE}\left(P_s^r, \boldsymbol{y}_s^r\right)$$

$$\mathcal{L}_e = \frac{1}{|\mathcal{R}| \times |\mathcal{S}|} \sum_{r \in \mathcal{R}} \mathrm{CE}\left(P_e^r, \boldsymbol{y}_e^r\right)$$

$$MRC = FCN(loss_{GAT} + loss_{BERT})$$

This iterates until all arguments are identified. There are no new arguments of high confidence in the two rounds, and we simultaneously identify all remaining arguments. In this way, the recognition result of argument with low confidence can avoid affecting the recognition effect of other arguments, and the purpose of reducing negative information transmission can be achieved.

4 Experiments and Results

We conducted extensive experiments to validate the advantages of the proposed event extraction framework, and the proposed sequence question generation strategy, and sequence label alignment mechanism.

Settings. The proposed event extraction method is evaluated in two aspects: one is the event detection performance, and the other is the accuracy of event extraction.

Datasets. The proposed extraction method along with the compared approaches are tested on both ACE 2005 and TAC KBP 2015 datasets. **Automatic Content Extraction (ACE) 2005.** It contains a complete set of training data in English, Arabic, and Chinese for the ACE 2005 technology evaluation. It contains 599 documents, which are annotated with 8 event types, 33 event subtypes, and 35 argument roles. **Text Analysis Conference Knowledge base Filling (TAC KBP) 2015.** The goal of TAC KBP event tracking is to extract information about the event so that it is suitable for input into the knowledge base.

Comparisons. We compare our extraction method with seven event extraction methods: **DBRNN** [Sha et al.,] is an LSTM-based framework that leverages the dependency graph information to extract event triggers and argument roles. **Joint3EE** [Nguyen and Nguyen, 2019] is a multi-task model that performs entity recognition, trigger detection and argument role assignment by shared Bi-GRU hidden representations. It is a neural model that achieves state-of-the-art performance on ACE 2005. **GAIL** [Zhang et al., 2019] is an ELMo-based model that utilizes generative adversarial network to help the model focus on harder-to-detect events. **DYGIE++** [Wadden et al., 2019] is a BERT-based framework that models text spans and captures within-sentence and cross-sentence context. These two models represent the state-of-the-art performance for feature-based and neural models, respectively.

4.1 Experimental Results on ACE 2005

We fine-tune the model on ACE 2005. While keeping other hyperparameters unchanged, we set the learning rate to 1×10^5 and the number of training epochs to 8. During fine-tuning, we employ negative sampling and set the negative sampling rate to 30%. In addition to fine-tuning on the full training set of ACE 2005, we consider a single-genre "partial"training setting in which the model is trained only on the 58 documents that appear in the newswire portion of the full training set.

Table 1 reports the performance of the systems on the four evaluation metrics. Training on the full training set improves F1 performance over training on the partial training set, giving the largest improvement on trigger identification and classification. Our model also tends to have higher recall than precision, especially on trigger identification and classification, and suffers from low precision compared to prior work. Our model achieves state-of-the-art performance on trigger identification and trigger classification.

Because our model does not explicitly incorporate entity mention detection, we hypothesize that our MRC-inspired approach predicts answer spans that are semantically correct but do not exactly match the gold answers, hurting performance on argument-related subtasks. We compare predicted arguments with gold references and find the following sources of errors:

- Relative clauses: Our model predicts Mosul whereas the gold answer is Mosul, where U.S.troops killed 17 people in clashes earlier in the week.
- Counts: The gold annotation is 300 billion yen but our model predicts 300 billion.
- Durations: The gold annotation is lasted two hours but our model predicts two hours.

4.2 Influence on number of training data

In order to verify that our model can achieve good performance even with little annotation data. We tested the performance of the model by changing the amount of training data, as shown in Table 4.

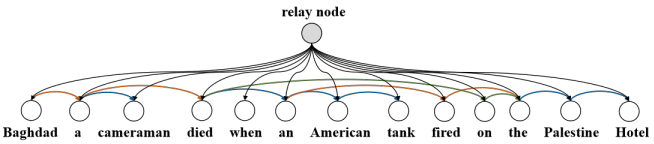

Figure 4: Influence on number of training data.

4.3 Error Analysis

By comparing gold and predicted labels for events and temporal relations and examining predicted probabilities for events, we identified three major sources of mistakes made by our structured model, as illustrated in Table7 with examples.

We further conduct error analysis and provide a number of representative examples. Table 7 summarizes error statistics for trigger detection and argument extraction. For event triggers, the majority of the errors relates to missing/spurious predictions and only 8.29% involves misclassified event types (e.g., a ELECT event is mistaken for a STARTPOSITION event). For event arguments, on the sentences that comes with at least one event in gold data, our framework extracts more argument spans only around 14% of the cases. Most of the time(54.37%), our framework extracts less argument spans, this corresponds to the results in Table 3, where the precision of our models are higher. In around 30% of the cases, our framework extracts same number of argument spans as in the gold data, half of them match exactly the gold arguments.

After examining the examples, we find the reasons for errors can be mainly divided into three categories: (1) Lack of knowledge for obtaining exact boundary for argument span. For example, in "Negotiations between Washington and Pyongyang on their nuclear dispute have been set for April23 in Beijing ...", for the ENTITY role, two argument spans should be extracted ("Washington" and"Pyongyang"). While our framework predicts the entire "Washington and Pyongyang" as the argument span. Although there's an overlap between the prediction and gold-data, the model gets no credit for it. (2) Lack of reasoning with document-level context. In sentence "MCI must now seize additional assets owned by

Ebbers, to secure the loan." There is a TRANSFER-MONEY event triggered by loan, with MCI being the GIVER and Ebbers being the RECIPIENT. In the previous paragraph, it's mentioned that "Ebbers failed to make repayment of certain amount of money on the loan from MCI." Without this context, it is hard to determine that Ebbers should be the recipient of the loan. (3) Data and lexical sparsity. In the following two examples, our model fails to detect the triggers of type END-POSITION. "Minister Tony Blair said ousting Saddam Hussein now was key to solving similar crises." "There's no indication if Erdogan would purge officials who opposed letting in the troops." It's partially due to they were not seen during training as trigger words. "ousting" a rare word and is not in the tokenizers' vocabulary. Purely inferring from the sentence context is hard for the purpose.

4.4 Ablation Study

To demonstrate key designs of Doc2EDAG, we conduct ablation tests by evaluating four variants: 1)-PathMem, removing the memory mechanism used during the EDAG generation, s 2)-SchSamp, dropping the scheduled sampling strategy during training, 3)-DocEnc, removing the Transformer module used for document-level entity encoding, and 4)-NegCW, keeping the negative class weight as 1 when doing path-expanding classifications. From Table 5, we can observe that 1) the memory mechanism is of prime importance, as removing it can result in the most drastic performance declines, over 10 F1 scores on four event types except for the ER type whose MER is very low on the test set; 2) the scheduled sampling strategy that alleviates the mis-match of entity candidates for event table filling between training and inference also contributes greatly, improving by 5 F1 scores on average; 3) the document-level entity encoding that enhances global entity representations contributes 2.1 F1 scores on average; 4) the larger negative class weight to penalize false positive path expanding can also make slight but stable contributions for all event types.

5 Conclusion and Future Works

We present an approach to event extraction that uses bleached statements to give a model access to information contained in annotation manuals. Our model incrementally refines the statements with values extracted from text. We also demonstrate the feasibility of making predictions on event types seen rarely or not at all. Future work can apply our approach to n-ary relation extraction.

6 Acknowledgements

We thank the anonymous reviewers for their insightful comments and suggestions. We also would like to thank Yu Hong for providing the ACE2005 corpus. Work was done when the first author was PhD candidate in Beihang University. Jianxin Li is the corresponding author.

References

[Chen et al., 2019] Chen, Y., Chen, T., Ebner, S., and Durme, B. V. (2019). Reading the manual: Event extraction as definition comprehension. *CoRR*, abs/1912.01586.

[Chen et al.,] Chen, Y., Xu, L., Liu, K., Zeng, D., and Zhao, J. Event extraction via dynamic multi-pooling convolutional neural networks. In *ACL, 2015*.

[Devlin et al.,] Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL, 2019*.

[Doddington et al.,] Doddington, G. R., Mitchell, A., Przybocki, M. A., Ramshaw, L. A., Strassel, S. M., and Weischedel, R. M. The automatic content extraction (ACE) program - tasks, data, and evaluation. In *LREC, 2004*.

[Du and Cardie, 2020] 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 2020, Online, November 16-20, 2020*, pages 671–683.

[Gui et al.,] Gui, T., Zou, Y., Zhang, Q., Peng, M., Fu, J., Wei, Z., and Huang, X. A lexicon-based graph neural network for chinese NER. In *EMNLP-IJCNLP, 2019*.

[Huang et al.,] Huang, L., Ji, H., Cho, K., Dagan, I., Riedel, S., and Voss, C. R. Zero-shot transfer learning for event extraction. In *ACL, 2018*.

[Lan et al.,] Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. ALBERT: A lite BERT for self-supervised learning of language representations. In *ICLR, 2020*.

[Li et al., 2020] Li, F., Peng, W., Chen, Y., Wang, Q., Pan, L., Lyu, Y., and Zhu, Y. (2020). Event extraction as multi-turn question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, Online Event, 16-20 November 2020*, pages 829–838.

[Liu et al., 2018] Liu, X., Luo, Z., and Huang, H. (2018). Jointly multiple events extraction via attention-based graph information aggregation. In *EMNLP, 2018*.

[Liu et al.,] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

[Nguyen and Grishman, a] Nguyen, T. H. and Grishman, R. Event detection and domain adaptation with convolutional neural networks. In *ACL, 2015*.

[Nguyen and Grishman, b] Nguyen, T. H. and Grishman, R. Modeling skip-grams for event detection with convolutional neural networks. In *EMNLP, 2016*.

[Nguyen and Nguyen, 2019] Nguyen, T. M. and Nguyen, T. H. (2019). One for all: Neural joint modeling of entities and events. In *AAAI, 2019*.

[Peters et al.,] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. Deep contextualized word representations. In *NAACL, 2018*.

Table 2: Ablation Study on Global Constraints

Model	EC			AI			ARC		
	P	R	F1	P	R	F1	P	R	F1
BERT(ours)	80.17	73.12	76.1	-	-	-	60.24	54.12	57.34
+GAT	80.35	73.23	76.14	-	-	-	60.36	54.47	58.31
+SAE	81.2	74.62	78.56	-	-	-	61.48	54.43	58.32
+SQG	-	-	-	-	-	-	-	-	-

[Sha et al.,] Sha, L., Qian, F., Chang, B., and Sui, Z. Jointly extracting event triggers and arguments by dependency-bridge RNN and tensor-based argument interaction. In *AAAI, 18*.

[Velickovic et al.,] Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *ICLR, 2018*.

[Wadden et al., 2019] Wadden, D., Wennberg, U., Luan, Y., and Hajishirzi, H. (2019). Entity, relation, and event extraction with contextualized span representations. In *EMNLP-IJCNLP, 2019*.

[Yang et al., 2019] Yang, S., Feng, D., Qiao, L., Kan, Z., and Li, D. (2019). Exploring pre-trained language models for event extraction and generation. In *ACL, 2019*.

[Yang et al.,] Yang, Z., Dai, Z., Yang, Y., Carbonell, J. G., Salakhutdinov, R., and Le, Q. V. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS, 2019*.

[Zhang et al., 2019] Zhang, T., Ji, H., and Sil, A. (2019). Joint entity and event extraction with generative adversarial imitation learning. *Data Intell. 2019*.