Event Detection via Derangement Question Answering

Anonymous ACL submission

Abstract

Event detection (ED), aiming to detect events from texts and categorize them, is vital to understanding the messages. Recently, ED without triggers has been proposed and gained benefits since it relieves the tedious effort of data labeling. However, it still suffers from several formidable challenges: multi-label, insufficient clues, and imbalanced event types. We, therefore, propose a novel Derangement Question-Answering (DQA) framework on top of BERT to tackle the above challenges. More specially, we treat the input text as a *question* and directly concatenate it with all event types, who are deemed as *answers*. Thus, by utilizing the original information, we can facilitate the power of self-attention in BERT to absorb the semantic relation between the original input text and the event types. Moreover, we design a simple yet effective *derangement* mechanism to relieve the issue of imbalanced event types. By including such perturbation, we can train a more robust model to promote the semantic information in the major events while recording the position of the minor events than the vanilla QA framework. The empirical results show that: (1) our proposed DQA framework attains state-of-the-art performance over previous competitive models. (2) Our model can automatically link the triggers with the event types while signifying the corresponding arguments.

1 Introduction

Event detection (ED), aiming to spot the appearance of predefined event types from texts and classify them, is an important step towards understanding the message (Edouard, 2017). Taking a sentence from ACE (Automatic Context Extraction):

> **S:** And they sent him to Baghdad and killed.

This sentence consists of two events, *Transport* and *Die*. A capable event detection system should correctly identify these two events simultaneously. At first glance, this task can be arduous because event types implicitly exist in one sentence.

In the literature, researchers first mainly tackle this problem by recognizing the event triggers and classifying the events accordingly (Li et al., 2013; Chen et al., 2015). An event trigger is a word or phrase that gives the *most clear* indication of an event occurrence. For example, in the above example, "sent" and "killed" are the event triggers for the events of *Transport* and *Die*, respectively. Various methods have been proposed to exploit event triggers for event detection, such as extracting syntactic, discourse, and other hand-engineered features as inputs for structured prediction (Li et al., 2013; Yang and Mitchell, 2016; Liu et al., 2018b) and neural architecture for joint tasks optimization (Nguyen et al., 2016; Nguyen and Nguyen, 2019; Wadden et al., 2019; Liu et al., 2018a). However, these methods usually require tedious manual effort on annotating both triggers and event types for training. After discovering event triggers are nonessential to event detection, a Type-aware Bias Neural Network with Attention Mechanisms (TBNNAM) has been explored to detect events *without* triggers (Liu et al., 2019).

In this paper, we focus on event detection without triggers due to the light need of data labeling. Especially, we aim at tackling the following formidable challenges: (1) **Multi-label issue:** Each input sentence may hold zero or multiple events, which can be formulated into a challenging machine learning task, or multi-label classification task. (2) **Insufficient clues:** It is observed that triggers are of significance to attain good performance on event detection (Zhang et al., 2020; Ebner et al., 2020). Without explicit triggers, we lack sufficient clues to identify the event types and need to dig deeply to promote the connection between the keywords and the corresponding event types. (3) **Imbalanced event distribution:** As shown in Fig. 2,

the events may follow the Matthew effect: some events dominate in the data while other events may contain only several pieces of samples. The imbalanced event distribution brings more obstacles in identifying the minor events.

To tackle the above challenges, we propose a Derangement Question Answering (DQA) framework to learn the semantic relation between the input texts and the event types. Figure 1 illustrates our proposed framework with three main modules: a QA encoder, the event derangement module (EDM), and the multi-label classifier. In the QA encoder, an input sentence is deemed as a "Question" and all event types are set as "Answers". Following by the "[CLS]" token, they are concatenated with the "[SEP]" token and fed as the input of BERT. Our setup is different from existing QA architectures (Du and Cardie, 2020; Liu et al., 2020) and seems more elegant because we only keep the original information and do not introduce extra tokens. The semantic relation between the input texts and event types are then learned by the powerful self-attention mechanism in BERT (Devlin et al., 2019). In EDM, when the target (or ground truth) event is a major event, we deliver the derangement procedure with probability q. That is, only r other events (possibly major events) are selected and shuffled. By appending such perturbation, we can train a more robust model to promote the semantic information in the major events while recording the position information of the minor events. The learned representations are then fed into a multi-label classifier to produce the final prediction. Our development is more efficient than (Liu et al., 2019).

In summary, the contribution of our work is threefold: (1) To the best of our knowledge, this is the first work to explore the QA framework for multi-label event detection without triggers. It utilizes BERT to directly learn the semantic relation between input texts and event types without introducing extra tokens or discarding any original information. (2) The proposed derangement mechanism is simple yet effective in resolving the imbalanced event distribution issue. Furthermore, it can enhance representation learning by promoting the semantic information in the major events and recording the position in the minor events. (3) We report state-of-the-art performance on the benchmark dataset. Our model also demonstrates the power of linking the triggers with the event types and simultaneously signifying the related arguments.

2 Related Work

Event Detection More recent research has focused on jointly extracting triggers and arguments for event detection. For example, in (Nguyen et al., 2016) and (Li et al., 2013), triggers and arguments have been extracted by bidirectional recurrent neural networks and structured Perceptron, respectively. In (Zhang et al., 2019), reinforcement learning has been deployed with generative adversarial networks for entity and event detection. Furthermore, with the success of the attention mechanism, many approaches have tried to integrate this mechanism into the proposed models. For example, in (Liu et al., 2018b), syntactic contextual representations have been learned by graph convolutional networks to extract triggers and arguments jointly by self-attention. In (Wadden et al., 2019), a BERT-based model has been proposed to learn multiple tasks, including named-entity recognition, relation extraction, and event extraction.

Conventional methods require time-consuming annotation of triggers, limiting the application of these approaches to scenarios without abundant labeled data. Therefore, researchers explore other methods to detect events without triggers. For example, a Type-aware Bias Neural Network with Attention Mechanisms (TBNNAM) (Liu et al., 2019) has been proposed by utilizing the attention mechanism. However, it still contains several insufficiency: (1) It turns the problem of event detection into a binary classification problem, which results in the domination of negative classes. (2) It does not consider the imbalanced event distribution issue. (3) It relies on the traditional LSTM and does not utilize more powerful pre-trained language models. These may limit the ability of TBNNAM to solve the task.

Question Answering With powerful pre-trained language models (PLMs) as encoders, we can simply represent the questions and passages by PLMs and get the answer by predicting the start and end index for a Machine Reading Comprehension (MRC) problem. Several pieces of work (Boros et al., 2021; Du and Cardie, 2020; Liu et al., 2020) have formulated the event detection task as a MRC or Question Answering (QA) task by generating questions for event extraction. However, they need to design suitable questions specifically and introduce extra tokens, which may disturb the model

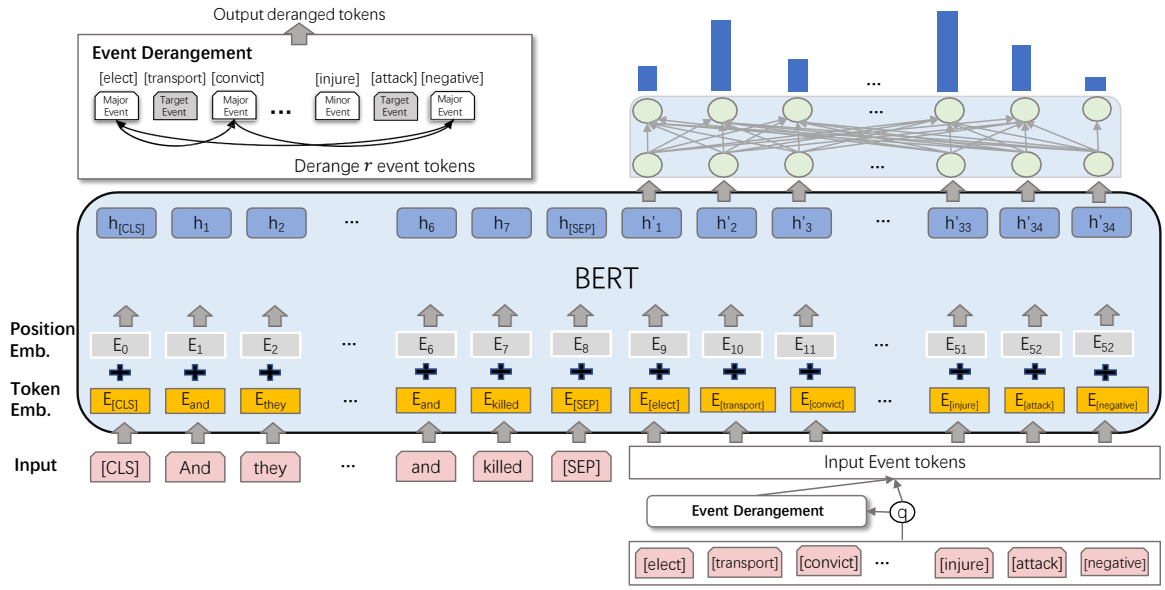


Figure 1: Our proposed DQA is on top of BERT. It consists of three main modules: QA encoder, the event derangement module (ETM), and the multi-label classifier. The ETM is amplified in the upper-left corner for better illustration; see more description in the main text.

learning.

3 Methodology

3.1 Task Definition

Following previous work (Ahn, 2006; Ji and Grishman, 2008; Liu et al., 2019), we are given a set of training data, $\{(x_i, y_i)\}_{i=1}^{N}$, where N is the number of sentence-event pairs. $x_i = w_{i1}w_{i2}\dots w_{i|x_i|}$ is the i-th sentence consisting of $|x_i|$ tokens and $y_i \subseteq \mathcal{S}$ is an event set, which records the corresponding event type(s). $\mathcal{S} = \{e_1, e_2, \dots, e_n\}$ consists of all n events. Here, we also add an additional label "negative" to specify those sentences that do not contain any events. Our goal is to train a model to detect the corresponding event type(s) as accurate as possible given an input sentence. This can be formulated as the multi-label classification task in machine learning. Our tasks here lie in (1) how to learn more precise representations to embed the semantic information between texts and event types? (2) How to deliver the multi-task classification task effectively?

Major Events vs. Minor Events Imbalanced event distribution is a major issue that we are targeted. Traditionally, Imbalance Ratio (IR) (Galar et al., 2012) is a typical metric to estimate the imbalance of the data. However, IR provides little information about the classes in the middle classes (Ortigosa-Hernández et al., 2017). Due to non-standard definition of major classes and minor classes in the multi-class cases and the simple setting in (Dong et al., 2018), we borrow its definition to distinguish the major events and the minor events. We first define a sorted sequence of all event types with respect to the number of instances in each class in the descending order:

$$S_{\text{SA}} = e_1 \dots e_n, \quad \text{where } |e_i| \geq |e_{i+1}|. \tag{1}$$

Here, e_i represents the i-th event type with $|e_i|$ instances.

Then, we define the set of major events as the top-k elements in S_{SA} while the remaining elements as the minor events:

$$E_{\text{Major}} = \{e_i \mid i = 1, 2, ...k\}, \tag{2}$$
$$E_{\text{Minor}} = \{e_i \mid i = k+1, ...n\}. \tag{3}$$

Now, we set α to indicate the percentage of the major events in all N sentence-event pairs:

$$\alpha * N = \sum_{i=1}^{k} |e_i|. \tag{4}$$

This definition will be used in our proposed *derangement* procedure.

3.2 Our Proposal

Figure 1 outlines the overall structure of our proposed DQA, which consists of three main modules: (1) a QA encoder, (2) the event derangement module (EDM), and (3) the multi-label classifier.

QA Encoder Our proposed DQA is based on BERT due to its power in learning the contextual representation in the sequence of tokens (Devlin et al., 2019). We then follow the standard BERT-style format and treat the original sentence as the **<question>** with all event types as the **<answers>**:

[CLS] <question> [SEP] <answers>

It is noted that our setting is different from existing QA models (Liu et al., 2020; Du and Cardie, 2020), which will generate extra tokens or relevant questions to describe an event. Our design tries to maintain the original texts as much as possible and let BERT automatically learn the semantic information between texts and event types.

Algorithm 1 Event Derangement

Require: Input sentence x; The initial event sequence S_{init}; The descending sorted sequence of all event types S_{SA}; Possibility q; Number r
Ensure: Deranged sequence of event tokens S_O

1: Initialize E_{GT} to the set of the ground truth event types implied by x
2: Initialize E_D with r elements in the beginning of S_{SA} that are not in E_{GT}
3: Initialize $E_{tmp} = \emptyset$ as a helper set
4: Initialize $S_O = []$
5: Generate $rand$ uniformly from [0, 1]
6: **if** $E_{GT} \cap E_{\text{Major}} \neq \emptyset$ and $rand < q$ **then**
7: **for** e_{curr} in S_{init} **do**
8: **if** e_{curr} in E_D **then**
9: Randomly select e from E_D and $e \neq e_{curr}$ and $e \notin E_{tmp}$
10: Append e to S_O
11: Add e to E_{tmp}
12: **else**
13: Append e_{curr} to S_O
14: **end if**
15: **end for**
16: **else**
17: $S_O = S_{\text{init}}$
18: **end if**
19: Return S_O

Specifically, given a training set, we first generate an event index $I_{\text{init}} = s_1 \dots s_n$, which is a permutation of $\{1, \dots, n\}$, and obtain its event sequence $S_{\text{init}} = e_{s_1} \dots e_{s_n}$. We need this sequence because we will fix it for the **<answer>** part during inference. Hence, given a sentence $x = w_1 \dots w_{|x|}$, we produce the input for our DQA as:

$$\text{Input} = [\text{CLS}]\ w_1\ \dots\ w_{|x|}\ [\text{SEP}]\ e_{s_1}\ \dots\ e_{s_n}. \quad (5)$$

Via BERT, we learn the hidden representations:

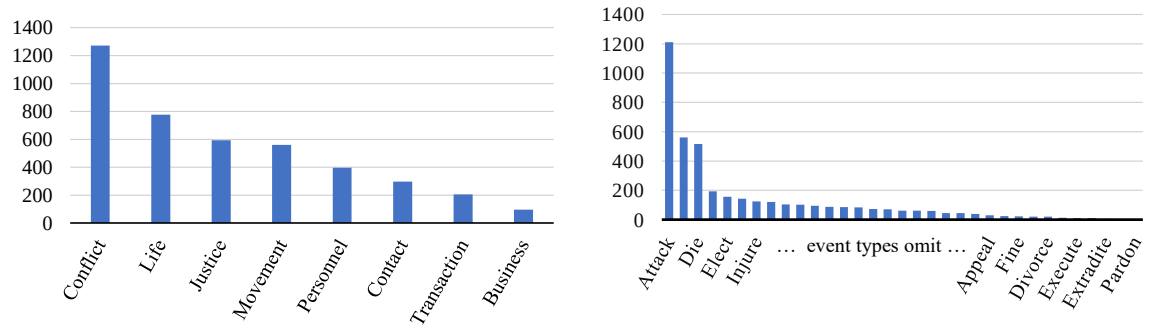
$$h_{[\text{CLS}]}, h_1^w, \dots, h_{|x|}^w, h_{[\text{SEP}]}, h_1^e, \dots, h_n^e = \text{BERT}(\text{Input}), \quad (6)$$

where h_i^w is the hidden state of the i-th input token and h_i^e is the hidden state of the corresponding event type, namely e_{s_i}. In the implementation, we treat e_i as a new word by placing a square bracket around it, i.e., the event *Transport* is converting to "[Transport]". This allows us to enrich the event tokens in the original BERT vocabulary and yield better performance in the evaluation; see more analysis in Appendix. A.1.

EDM Since event types are converted as input tokens of BERT, it raises a critical question: whether we need to add the position embeddings on the event type during training? An observation in (Pham et al., 2021) shows that BERT is position-insensitive and can attain more similarity on the Quora QQP task when shuffling some words. However, in our test, we discover that position embeddings can also provide hints for distinguishing the event types; see more verification in Sec. 5.2. We conjecture that in the imbalance data, our proposed QA framework tends to memorize the position information rather the semantic information.

In order to alleviate this effect, we introduce the *derangement* mechanism on the event tokens to add perturbation during training. Derangement is a classical permutation term in combinatorics, where a permutation of the elements of a set makes no element appear in its original position. We conduct the derangement procedure only when the target (or the ground truth) event is a major event as defined in Eq. (4). After that, we only deliver it with probability q. This is similar to the procedure of Masked LM in BERT and allows us to balance the position memorizing and semantic information absorption. In implementation, we randomly select r other (usually major) events from E_{SA} except the target event for derangement. This procedure is summarized in Algorithm 1 and yields a deranged sequence with the same size of n for training. In Algorithm 1,

- In line 2, E_D selects r events, which is not in E_{GT} and whose size is relatively large, i.e., in the beginning of S_{SA}.
- In line 5, the procedure is conducted only when the target event is a major event and is performed with probability of q.



(a) Event main types

(b) Event subtypes

Figure 2: The distribution of event main types and event subtypes on the ACE2005 training data.

- In lines of 6-14, usually the major events are deranged because the elements in E_D are selected from the top elements in the sorted sequence, S_{SA}.

Multi-label Classifier After we learn the contextualized representations of x and S_{init} from the proposed QA structure, we turn to construct the multi-label classifier. Different from traditional method to encode the [CLS] token, we feed the event tokens to a Multi-Layer Perception (MLP) for classification because we can obtain better performance; see more supporting results in Appendix A.2. Hence, we compute the predicted probability of an input sentence x to the corresponding events by

$$\hat{p} = \text{MLP}\left(h_1^e, ..., h_n^e\right). \tag{7}$$

During inference, we determine the event labels when $\hat{p} \geq 0.5$.

The model parameters of our DQA can then be attained by minimizing the following loss function:

$$\mathcal{L} \propto -\sum_{i=1}^{N}\sum_{j=1}^{n}(p_{ij}\log(\hat{p}_{ij})+(1-p_{ij})\log(1-\hat{p}_{ij})) \tag{8}$$

where $p_{ij} = 1$ represents the corresponding event for the i-th input text. Different from (Liu et al., 2019), which converts the multi-label classification task into a binary classification task, our proposed DQA can directly train the model to output the multi-label results simultaneously.

4 Experiments

4.1 Experimental Setups

Dataset and Evaluation We conducted experiments on the ACE2005 English corpus. The ACE2005 corpus consists of 8 event main types and 33 subtypes. As shown in Fig. 2, the corpus follows the imbalanced event distribution and is more imbalanced (IR≈605.5) for the event subtypes than that (IR≈13.1) in the event main types. For example, the types of *Attack*, *Transport*, and *Die* account for over half of the total training data. For fair comparison, we follow the evaluation of (Li et al., 2013; Liu et al., 2019, 2020), i.e., randomly selecting 30 articles from different genres as the validation set, subsequently delivering a blind test on a separate set of 40 ACE2005 newswire documents, and using the remaining 529 articles as the training set. The standard metrics: Precision (P), Recall (R), and F1 scores (F1), are applied to evaluate the model performance.

Implementation Details Our implementation is in PyTorch [1]. The BERT base model (uncased) from Hugging Face (Wolf et al., 2019) is adopted as the backbone model, which consists of 12 layers, 768 hidden units, and 12 attention heads. The MLP consists of two layers with the hidden size being 768 and yields an output of 34 dimension to predict the probability of the input sentence assigned to the corresponding 34 classes. We follow (Dong et al., 2018) and set α to 0.5. In EDM, q is 0.2 and r is 24 from empirical evaluation. The batch size is 8. The learning rate is set as 2×10^{-5}. The dropout rate is 0.1. ADAM is the optimizer (Kingma and Ba, 2015). We train our models for 10 epochs to give the best performance. All experiments are conducted on a NVIDIA A100 GPU.

[1] https://www.dropbox.com/s/wqp1u1gu6bqs6da/DQA.zip?dl=0

Methods	Subtypes (%)			Main (%)		
	P	**R**	**F1**	**P**	**R**	**F1**
TBNNAM (Liu et al., 2019)	76.2	64.5	69.9	-	-	-
DYGIE++, BERT + LSTM (Wadden et al., 2019)	-	-	68.9	-	-	-
DYGIE++, BERT Finetune (Wadden et al., 2019)	-	-	69.7	-	-	-
BERT_QA_Trigger (Du and Cardie, 2020)	71.7	73.7	72.3	-	-	-
DMBERT (Wang et al., 2019)	77.6	71.8	74.6	-	-	-
RCEE_ER (Liu et al., 2020)	75.6	74.2	74.9	-	-	-
DMBERT + Boot (Wang et al., 2019)	77.9	72.5	75.1	-	-	-
BERT Finetune	72.8	68.7	70.7	78.0	70.8	74.2
Our BERT_QA	76.9	72.3	74.7	78.9	75.4	77.1
Our BERT_DQA	**79.5**	**76.8**	**78.1**	**78.7**	**79.0**	**78.9**

Table 1: Event detection results on both the event subtypes and event main types of the ACE2005 corpus.

4.2 Overall Performance

We compare our proposed BERT_QA and BERT_DQA with several competitive baselines: **TBNNAM** (Liu et al., 2019): an LSTM model detecting events without triggers, and BERT-based models for both trigger detection and event detection: **DYGIE++** (Wadden et al., 2019): a BERT-based framework modeling text spans; **BERT_QA_Trigger** (Du and Cardie, 2020) and **RCEE_ER** (Liu et al., 2020): both BERT-based models converting event extraction into a QA task; **DMBERT** (Wang et al., 2019): a BERT-based model leveraging adversarial training for weakly supervised events, where DMBERT Boot stands for bootstrapped DMBERT.

Table 1 reports the overall performance on the ACE2005 corpus. It shows that (1) previous models only evaluate the performance on the event subtypes. Although our proposed BERT_QA does not access to the triggers, it attains significant better performance than TBNNAM, DYGIE++, and BERT_QA_Trigger. Its performance is also competitive to DMBERT and RCEE_ER, with 74.7% F1 score, only 0.4% less F1 score than that in the best baseline, DMBERT Boot. The result shows that our proposed QA framework is effective to learn the semantic information between given texts and event types. (2) After introducing the derangement mechanism, our proposed BERT_DQA can significantly outperform all compared methods in all three metrics. Especially, it attains 3.0% more F1 score than the best baseline. (3) To verify the generalization of our proposal, we also conduct experiments to evaluate the performance on event main types. The setting of the model parameter is the same as that on the event subtypes, except $r = 3$ for DQA. The results show that our proposed BERT_QA and BERT_DQA gain further improvement, i.e., 2.9% and 4.7% F1 score over the finetuned BERT, respectively. The results show the consistence of our proposal and it seems that BERT_DQA can attain better performance when the dataset (the event subtypes) is more imbalanced; see more supporting results in Appendix A.3.

Position Embedding	P	R	F1
Fixed	75.7	71.6	73.6
Learnable	76.9	72.3	74.7

Table 2: Evaluation results by applying fixed or learnable position embeddings on the same event sequence.

	P	R	F1
BERT_QA_Shuffle_Both	68.2	70.6	66.4
BERT_QA	76.9	72.3	74.7
BERT_QA_Shuffle_Test	18.2	9.2	12.2
BERT_DQA_Shuffle_Test	66.0	45.1	53.6
BERT_DQA	**79.5**	**76.8**	**78.1**

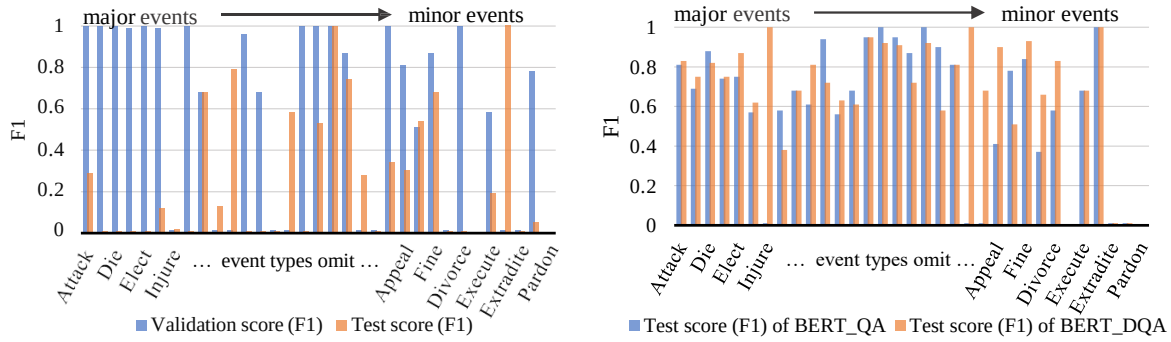
Table 3: Evaluation results on different event sequences.

5 More Analysis

We conduct more detailed analysis to verify the effect of our proposal.

5.1 Effect of Position Embeddings

We test the effect of the position embeddings on our proposed BERT_QA. Here, we test two cases, fixed position embeddings and learnable position embeddings for the given event sequence. Results



(a) $q = 1.0$ (b) $q = 0.2$

Figure 3: Effect of EDM: In Fig. 3(a), the blue color denotes the F1 score on the validation set and the red color for the test set. When evaluating the model at validation set during training, event derangement is applied in this case. In Fig. 3(b), the blue color denotes the F1 score of BERT_QA on the test set and the red color for BERT_DQA. For better illustration, we only show parts of events and set the label segmentation to 2.

in Table 2 show that BERT_QA is a position-aware: via a learnable position embedding, it can gain better performance, around 1% improvement on the F1 score.

5.2 Effect of Event Sequences

Table 3 reports the results of BERT_QA and BERT_DQA with different event sequences. Here, "BERT_QA_Shuffle_Both" stands for a BERT_QA trained and tested on randomly shuffled event sequences. It can be deemed as a baseline of BERT_QA because it only absorbs the semantic information between texts and event types while totally forgetting the position information. "BERT_QA_Shuffle_Test" defines a BERT_QA trained on a given event sequence while testing on a shuffled event sequence. "BERT_DQA_Shuffle_Test" tests the case of a BERT_QA trained on a given event sequence and the derangement mechanism while testing on a shuffled event sequence. The results show that (1) "BERT_QA_Shuffle_Both" attains satisfactory performance, which shows the power of our BERT_QA in absorbing the semantic information between texts and event types. (2) The performance of "BERT_QA_Shuffle_Test" drops significantly because BERT_QA is totally polluted and confused by the randomly generated event sequences when inference. The result implies that BERT_QA leverages position information to classify events other than semantic knowledge. (3) "BERT_DQA_Shuffle_Test" attains a closer performance to "BERT_QA_Shuffle_Both". This result shows that the derangement mechanism can effectively avoid BERT_QA from excessively memorizing positions and increase the model's learning of semantic knowledge. However, since we only shuffle parts of events, the semantic relationship is not fully absorbed as that in "BERT_QA_Shuffle_Both". (4) Overall, BERT_QA can attain good performance while BERT_DQA can further improve it and achieves the best performance.

5.3 Effect of EDM

Figure 3(a) shows the extreme case when $q = 1.0$, where EDM fails to identify major events but still recognizes the minor events during inference. An underlying observation is that during training, the target events remain in the original position while other events are deranged, this will disturb the information forwarding on the deranged events. Accordingly, it makes the target events stand out and can be easily recognized, as shown in the results of the validation set. Therefore, there is little loss on the target events during training, which prevents the model from learning information for the target events via backpropogation.

Figure 3(b) shows that setting a suitable q ($= 0.2$) can prevent the model's overfitting on major events while enhancing the recognition on the minor events. This is similar to under-sampling on the major events, which leads to a more balanced updating on BERT_DQA. Our EDM may echo the mechanism in response to sensory deprivation (Merabet and Pascual-Leone, 2010): neurons in human brain are reorganized to functioning regions, which, for instance, makes the blind have stronger hearing.

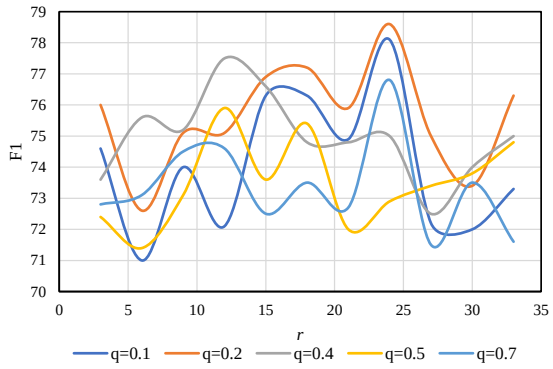


Figure 4: Effect of q and r.

5.4 Effect of q and r

We select q from $\{0.1,\ 0.2,\ 0.4,\ 0.5,\ 0.7\}$ and r from $\{3, 6, \ldots, 33\}$, i.e., equally dividing all event types into 10 buckets. We ignore larger q's because they usually fail the model on major events; see results when $q = 1.0$ in Fig. 3(a). Figure 4 shows the performance with respect to r for different q. It is shown that the best performance is attained when $q = 0.2$ and $r = 24$. The trends also show that a smaller q can usually yield better performance while r is selected in the range of 15 and 25 because r can indicate the scale of perturbation. A smaller r may cause negligible perturbation and a larger r may affect the disturbance of the minor events.

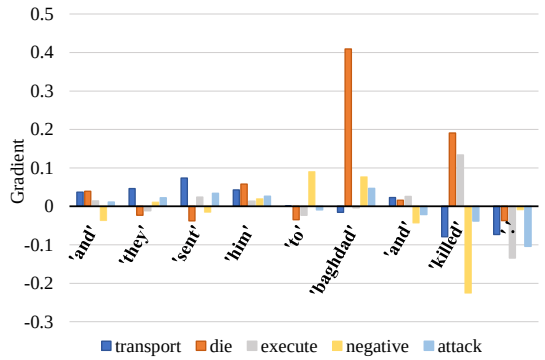


Figure 5: Gradient visualization of words in a sentence with respect to five typical event types; see more description in the main text.

5.5 Gradient Explanation

The gradient explanation is a more stable method to explain the model (Adebayo et al., 2018) than the attention weights in BERT because the attention weights may be misleading (Jain and Wallace, 2019) or are not directly interpretable (Brunner et al., 2020). We then compute the gradient with respect to the token embeddings, which quantifies the influence of changes in the tokens on the predictions. Here, we pick the example in Sec. 1 and select five typical events: "Die" and "Transport" are the target events; "Negative" and "Attack" are two common event types; and "Execute" is a minor event. Figure 5 clearly shows that (1) For the event of "Die", our BERT_DQA can automatically focus on its trigger word "killed" while for the event "Transport", the trigger "sent" is also noticed by model. But for non-target events, our BERT_DQA attains low gradients on the triggers or gets high gradients on unrelated tokens, such as "to" and ".". (2) More importantly, our BERT_DQA can surprisingly spot the related arguments for the events. For example, for the event of "Die", "Baghdad" yields a significant higher gradient, which corresponds to the argument of PLACE. Similarly, for the event of "Transport", "they" and "him" also yield relatively larger gradients, which exactly correspond to the argument of ARTIFACT and AGENT, respectively. The observations shows the power of our proposed DQA framework in not only linking triggers to the corresponding events, but also highlighting the event arguments, which is better than those traditional event extraction methods with only trigger extraction.

6 Conclusion and Future Work

In this paper, we propose a novel Derangement Question Answering (DQA) framework on top of BERT to detect events without triggers and under the imbalanced setting. By treating the input text as a question and directly concatenating it with all event types as answers, we utilize the power of self-attention in BERT to absorb the semantic relation between the original input text and the event type(s). Moreover, the proposed event derangement module is simple yet effective to relieve the imbalanced event types. By introducing such perturbation, we can train a more robust model than the vanilla BERT_QA framework. We conduct intensive evaluation and show that our proposed DQA framework attains state-of-the-art performance over previous methods and can automatically link the triggers with the event types while signifying the related arguments. In the future, we would like to test how to generate the optimal initial event sequence and adapt our proposal to other information extraction tasks to study its application scope.

References

Julius Adebayo, Justin Gilmer, Michael Muelly, Ian J. Goodfellow, Moritz Hardt, and Been Kim. 2018. Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 9525–9536.

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

Emanuela Boros, Jose G. Moreno, and Antoine Doucet. 2021. Event detection as question answering with entity information. *CoRR*, abs/2104.06969.

Gino Brunner, Yang Liu, Damian Pascual, Oliver Richter, Massimiliano Ciaramita, and Roger Wattenhofer. 2020. On identifiability in transformers. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. 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, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 167–176. The Association for Computer Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Qi Dong, Shaogang Gong, and Xiatian Zhu. 2018. Imbalanced deep learning by minority class incremental rectification. *IEEE transactions on pattern analysis and machine intelligence*, 41(6):1367–1381.

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 2020, Online, November 16-20, 2020*, pages 671–683. Association for Computational Linguistics.

Seth Ebner, Patrick Xia, Ryan Culkin, Kyle Rawlins, and Benjamin Van Durme. 2020. Multi-sentence argument linking. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 8057–8077. Association for Computational Linguistics.

Amosse Edouard. 2017. *Event detection and analysis on short text messages. (Détection d'événement et analyse des messages courts)*. Ph.D. thesis, University of Côte d'Azur, France.

Mikel Galar, Alberto Fernández, Edurne Barrenechea Tartas, Humberto Bustince Sola, and Francisco Herrera. 2012. A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. *IEEE Trans. Syst. Man Cybern. Part C*, 42(4):463–484.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 3543–3556. Association for Computational Linguistics.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *ACL 2008, Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, June 15-20, 2008, Columbus, Ohio, USA*, pages 254–262. The Association for Computer Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

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, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pages 73–82. The Association for Computer 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 2020, Online, November 16-20, 2020*, pages 1641–1651. Association for Computational Linguistics.

Jian Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2018a. Event detection via gated multilingual attention mechanism. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 4865–4872. AAAI Press.

Shulin Liu, Yang Li, Feng Zhang, Tao Yang, and Xinpeng Zhou. 2019. Event detection without triggers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,*

NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), pages 735–744. Association for Computational Linguistics.

Xiao Liu, Zhunchen Luo, and Heyan Huang. 2018b. Jointly multiple events extraction via attention-based graph information aggregation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 1247–1256. Association for Computational Linguistics.

Lotfi B Merabet and Alvaro Pascual-Leone. 2010. Neural reorganization following sensory loss: the opportunity of change. *Nature Reviews Neuroscience*, 11(1):44–52.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. 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, June 12-17, 2016*, pages 300–309. The Association for Computational Linguistics.

Trung Minh Nguyen and Thien Huu Nguyen. 2019. One for all: Neural joint modeling of entities and events. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 6851–6858. AAAI Press.

Jonathan Ortigosa-Hernández, Inaki Inza, and Jose A Lozano. 2017. Measuring the class-imbalance extent of multi-class problems. *Pattern Recognition Letters*, 98:32–38.

Thang Pham, Trung Bui, Long Mai, and Anh Nguyen. 2021. Out of order: How important is the sequential order of words in a sentence in natural language understanding tasks? In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1145–1160, Online. Association for Computational Linguistics.

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 2019, Hong Kong, China, November 3-7, 2019*, pages 5783–5788. Association for Computational Linguistics.

Xiaozhi Wang, Xu Han, Zhiyuan Liu, Maosong Sun, and Peng Li. 2019. Adversarial training for weakly supervised event detection. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 998–1008.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Bishan Yang and Tom M. Mitchell. 2016. 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, June 12-17, 2016*, pages 289–299. The Association for Computational Linguistics.

Tongtao Zhang, Heng Ji, and Avirup Sil. 2019. Joint entity and event extraction with generative adversarial imitation learning. *Data Intell.*, 1(2):99–120.

Zhisong Zhang, Xiang Kong, Zhengzhong Liu, Xuezhe Ma, and Eduard H. Hovy. 2020. A two-step approach for implicit event argument detection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7479–7485. Association for Computational Linguistics.

A Appendix

We provide more analysis to support our proposal.

Conversion	P	R	F1
Original	75.2	67.6	71.7
New	77.3	68.2	72.5

Table 4: Results of different conversion ways of event tokens.

A.1 Effect of Event Tokens Conversion

There are two intuitive ways to treat the event tokens in our proposed DQA framework. One is to treat them as old words in the BERT dictionary, so that we can initialize the event representations

by utilizing BERT's pre-trained word embeddings. The other way is to treat them as new words, so that we can learn the event representations from scratch. Hence, we can directly feed the *original* event words in the DQA framework or add a square bracket around the event words to convert them into *new* words, e.g., "Transport" to "[Transport]", in the BERT dictionary.

Table 4 reports the compared results and shows that converting event types into new words can attain substantial improvement in all three metrics than treating them as the original words in BERT dictionary. We conjecture that it may arise from WordPiece (Wu et al., 2016) in BERT implementation because BERT will separate an event word into several pieces when it is relatively long. This brings the difficulty in precisely absorbing the semantic relation between the words in input texts and event types. On the contrary, when we treat an event word as a new word, BERT will deem them as a whole. Though BERT learns the event representations from scratch, it is still helpful to establish the semantic relationship between words and event types.

A.2 Inputs for the Multi-label Classifier

There are two kinds of inputs for the multi-label classifier: the representation of the [CLS] token, or the event representations. We feed these two inputs into the same MLP to predict the probability of an input sentence x to the corresponding events.

Input	P	R	F1
[CLS]	77.3	68.2	72.5
All event tokens	76.9	72.3	74.7

Table 5: Results of different inputs for the multi-label classifier.

Table 5 reports the performance of different inputs for the multi-label classifier and shows that by feeding the event representations as the input, our BERT_QA can significantly improve the performance on Recall and the F1 score with competitive Precision score than only using the representation of the [CLS] token. We conjecture that the event representations have injected more information into the multi-label classifier than only using the representation of the [CLS] token.

A.3 Limitation of EDM

We conduct evaluation on a more balanced dataset to investigate the limitation of EDM. We first select seven relatively balance event types, yielding an imbalance ratio around 1.8, from the subtypes of the ACE2005 corpus; see the data distribution in Fig. 6. In the experiment, we set q to 0.2 and r to 6 for good performance on BERT_DQA.

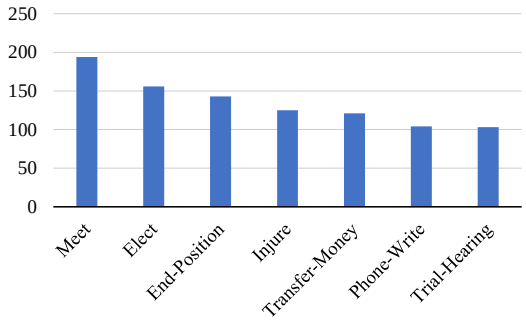


Figure 6: Data distribution of seven balanced event subtypes.

Model	P	R	F1
BERT_QA	76.4	77.8	77.1
BERT_DQA	75.0	76.3	75.6

Table 6: The performance of our BERT_QA and BERT_DQA on a more balanced dataset.

Table 6 reports the comparison results of BERT_QA and BERT_DQA and shows that BERT_QA attains satisfactory results and beats BERT_DQA in all three metrics. The results imply that the derangement procedure plays an important role when the dataset is more imbalanced. When the dataset is relatively balanced, we can turn to BERT_QA and attain good performance due to the power of self-attention in BERT.

A.4 Error Analysis

We conduct error analysis on test dataset in this section. There are three main kinds of errors:

– The main error comes from event misclassification, accounting for 52.9% of the total errors. The error also includes that BERT_DQA detects more event types than the ground truth. The most event type that BERT_DQA overpredicts is the event of *Attack*. A typical example is given below:

> **S:** The officials, who spoke on ... 26 words omitted ... on the U.S.-backed **war** resolution.

BERT_DQA deems this sentence belonging to the event of *Attack*, where the ground truth is

the event of *Meet*. This error is normal because the word "war" is a common trigger to the event of *Attack*, which yields BERT_DQA misclassifying it. In this dataset, the event of *Attack* is the most dominating event type, which makes it likely to classify the texts of other events as *Attack* when the texts hold some similar words to the triggers of *Attack*.

– The second type of errors is that BERT_DQA outputs fewer event types than the ground truth, which accounts for 28.9% of errors. The frequently missing event types are *Transfer-Money* and *Transfer-Ownership*. One typical example is

> **S:** Until Basra, U.S. and British troops ... 6 words omitted... they **seized** nearby Umm Qasr ... 3 words omitted... **secure** key oil fields.

BERT_DQA fails to identify the event of *Transfer-Ownership*, which is indicated by the trigger, "secure", while recognizing the event of *Attack*, implied by the trigger if "seized". On the one hand, the Imbalanced Ratio of *Attack* and *Transfer-Ownership* is 14.2. There are much fewer training data for BERT_DQA to learn the patterns of *Transfer-Ownership* than those of *Attack*. On the other hand, deeper semantic knowledge is needed for understanding the event of *Transfer-Ownership*, whose trigger words are more diverse and changeable. The triggers for *Transfer-Ownership* may include "sold", "acquire", and "bid", etc.

– The third type of errors lies in outputting none-event sentences. When there are no event types in a sentence, BERT_DQA may fail to classify it as the type of *negative*. This is because there is no sufficient clues for BERT_DQA to learn the patterns from the type of *negative*. BERT_DQA also turns out to give low predicted probabilities on all event types.