Abstract

Event detection (ED), aiming to detect events from texts and categorize them, is vital to understanding the actual happenings in real life. 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 mechanism on a machine Reading Comprehension (DRC) framework to tackle the above challenges. More specially, we treat the input text as Context and concatenate it with all event types that are deemed as Answers with an omitted default question. Thus, by appending input text and event types simultaneously, we can facilitate the power of self-attention in pre-trained language models, e.g., BERT, to absorb the semantic relation among them. Moreover, we design a simple yet effective derangement mechanism to relieve the imbalanced training. By introducing such perturbation mainly on major events, we can prohibit major events from excessive learning or implicitly under-sample the instances of the major events. This yields a more balanced training to resolve the imbalanced learning issue. The empirical results show that: (1) our proposed framework attains state-of-the-art performance over previous competitive models, and (2) by-product, our model can signify the connection of triggers and arguments to events for further analysis.

1 Introduction

The task of event detection (ED), aiming to spot the appearance of predefined event types from texts and classify them, is vital to understanding the actual happenings in real life (Edouard, 2017; Saeed et al., 2019). Take an example from ACE (Automatic Context Extraction):

S: And they sent him to Baghdad and killed.

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

In the literature, researchers usually tackle this problem via a two-stage trigger-based framework. That is, triggers (i.e., words or phrases providing the most clear indication of an event occurrence) are first identified and then events are recognized accordingly (Ahn, 2006; Li et al., 2013; Chen et al., 2015). For example, in the above example, “sent” and “killed” are the triggers for Transport and Die, respectively. Following this line, various methods have been proposed, including 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, trigger identification is an intermediate step for event detection and requires demanding effort on annotation. After discovering triggers are nonessential to event detection, trigger-free methods, e.g., the Type-aware Bias Neural Network with Attention Mechanisms (TBNNAM) (Liu et al., 2019), have been proposed.

In this paper, we focus on event detection without triggers due to the light need of data labeling. 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:** Triggers are of significance to attain good performance on event detection (Zhang et al., 2020; Ebner et al., 2020). Without explicitly including triggers, we may lack sufficient clues to identify the event types and need to seek alternatives to shed light on the correlation between words and
the event types. (3) **Imbalanced events distribution:** As shown in Fig. 2, events may follow the Matthew effect. That is, some events dominate the data while others contain only several instances. The imbalanced event distribution brings significant obstacles to detect minor events.

Hence, we propose a Derangement mechanism on a machine Reading Comprehension (DRC) framework to tackle the challenges. Figure 1 illustrates our proposed framework with three main modules: the RC encoder, the event derangement module (EDM), and the multi-label classifier. In the RC encoder, the input sentence, deemed as “Context”, and all event tokens, appended as “Answers”, are fed into BERT (Devlin et al., 2019) together. Such design allows the model to observe all available information without the need of explicitly indicating triggers and enables the model to automatically learn helpful semantic relations between input texts and event tokens through the self-attention mechanism of Transformer (Vaswani et al., 2017). During training, the EDM is activated only when the grand-truth event is a major event with a certain probability. By perturbing the order of other major event tokens, the model can prevent excessively updating the instances of major events, which implicitly under-samples the training instances of the major events and yields a more balanced training to resolve the imbalanced learning issue. Finally, the learned contextual representations of event tokens are fed into a multi-label classifier to produce the probabilities of the input text to each event type.

In summary, the contribution of our work is threefold: (1) We propose a competitive paradigm to an important task, namely multi-label event detection without triggers. Through a simplified machine reading comprehension framework, we can directly capture the semantic relation between input texts and event types without explicitly including triggers. (2) During training, we implement a simple yet effective mechanism, i.e., the derangement mechanism, to overcome the imbalanced training issue. By perturbing the order of major event tokens, we implicitly under-samples the training instances from the major events and fulfill a more balanced training. (3) We report that our proposal achieves the state-of-the-art performance at event detection on the public benchmark dataset. The results also exhibit the potential of our proposal to link the triggers to the corresponding events and simultaneously signify the necessary arguments.

### 2 Related Work

**Event Extraction (EE)** A major stream of approaches focus on event extraction to identify both triggers and arguments, which can be categorized as trigger-based approaches. For example, in (Li et al., 2013), structured Perceptron has been exploited on hand-made features to identify triggers and arguments. In (Nguyen et al., 2016), triggers and arguments are jointly identified by utilizing bidirectional recurrent neural networks. In (Zhang et al., 2019), reinforcement learning is deployed with generative adversarial networks for entity and event extraction. Furthermore, witnessing the success of attention mechanism, many approaches have tried to integrate attention into the proposed models. For example, in (Liu et al., 2018b), syntactic contextual representations are learned by graph convolutional networks to extract triggers by self-attention. In (Wadden et al., 2019), a BERT-based model was proposed to multi-task learning for named-entity recognition, relation extraction, and event extraction. Another stream of trigger-based approaches formulate the EE task as a machine reading comprehension or question answering task (Du and Cardie, 2020; Liu et al., 2020). For example, in (Du and Cardie, 2020), a predefined question template concatenating with the input sentence is fed into BERT to identify the corresponding triggers and arguments. In (Liu et al., 2020), similar framework is proposed with different template design. However, this kind of implementation has to search optimal results from multiple predefined templates during inference.

**Event Detection without Triggers** The above trigger-based methods heavily rely on manually annotated triggers, which is time-consuming. Therefore, researchers have explored alternative trigger-free methods for event detection. The Type-aware Bias Neural Network with Attention Mechanisms (TBNNAM) (Liu et al., 2019) has been proposed to utilize the attention mechanism to incorporate information of event types to input sentences. One shortcoming is that it turns a multi-label event detection into binary classification on each event, which can be tedious and inefficient during inference.
3 Methodology

3.1 Task Definition

Following (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. \( x_i = w_{i1}w_{i2}...w_{ik} \) is the \( i \)-th sentence with \( |x_i| \) tokens and \( y_i \subseteq S \) is an event set, which records all related event(s). \( S = \{e_1, e_2, ..., e_n\} \) consists of all \( n \) events, including an additional “negative” event meaning that sentences 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 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 in our setting. Traditionally, Imbalance Ratio (IR) (Galar et al., 2012) is a typical metric to estimate the imbalance of the data. However, IR provides little distribution information about the middle classes (Ortigosa-Hernández et al., 2017). To articulate the differences of major events and minor events, we borrow its definition in (Dong et al., 2018) to distinguish them. We first sort all event types in descending order with respect to the number of instances in each class and obtain the sorted sequence:

\[
S_{\text{MA}} = e_1 \ldots e_n, \quad \text{where} \quad |e_i| \geq |e_{i+1}|. \quad (1)
\]

Here, \( e_i \) denotes the \( i \)-th event type with \( |e_i| \) instances.

Then, we define the set of major events as the top-\( k \) elements in \( S_{\text{MA}} \) while the remaining elements as the minor events:

\[
E_{\text{Major}} = \{e_i \mid i = 1, 2, ..., k\}, \quad (2)
\]

\[
E_{\text{Minor}} = \{e_i \mid i = k + 1, ..., n\}, \quad (3)
\]

where \( k \) is determined by a hyperparamter of \( \alpha \) by rounding to the nearest integer if it is a float. Here, \( \alpha \) indicates the percentage of the major events in all \( N \) sentence-event pairs:

\[
\alpha \times N = \sum_{i=1}^{k} |e_i|. \quad (4)
\]

Usually, \( \alpha \) is simply set to 0.5 as (Dong et al., 2018).

3.2 Our Proposal

Figure 1 outlines the overall structure of our proposed DRC, which consists of three main modules: the RC encoder, the multi-label classifier, and the event derangement module (EDM) for training.
**RC Encoder** Our proposal is based on BERT due to its power in learning the contextual representation in the sequence of tokens (Devlin et al., 2019). We present a simplified machine reading comprehension (MRC) framework:

```
[CLS] Context [SEP] Answers
```

where Context is the input sentence and Answers sequence all the event types. This setup is close to MRC with the multiple choices option. That is, it views the input sentence as Context and event types as the multiple choices (or Answers) with an omitted default question: “What is the event type/what are the event types in the Context?” With both input texts and event tokens for the input of BERT, we can utilize BERT to automatically capture the relation between input texts and event types without explicitly indicating the triggers.

---

**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 $S_0$

1. Initialize $E_{GT}$ as the set of the ground truth event types implied by $x$
2. Initialize $E_0$ with $r$ events that are not in $E_{GT}$ from the beginning sequence of $S_{SA}$
3. Initialize $E_{tmp} = \emptyset$ # a helper set to record the selected event types during derangement
4. Initialize $S_0 = []$
5. Generate $rand$ uniformly from $[0, 1]$
6. if $E_{GT} \cap E_{tmp} \neq \emptyset$ and $rand < q$ then
   7. for $e_{curr}$ in $S_{init}$ do
      8. if $e_{curr}$ in $E_{GT}$ then
         9. Randomly select $e$ from $E_0$ and $e \neq e_{curr}$ and $e \notin E_{tmp}$
      10. Append $e$ to $S_0$
      11. Add $e$ to $E_{tmp}$
     12. else
        13. Append $e_{curr}$ to $S_0$
     14. end if
   15. end for
   else
   17. $S_0 = S_{init}$
   18. end if
19. Return $S_0$

---

In the implementation, given a training set, we first generate a random event order index $I_{init} = s_1 \ldots s_n$, which is a permutation of $\{1, \ldots, n\}$, and obtain its initial sequence of event tokens $S_{init} = e_{s_1} \ldots e_{s_n}$. Without specifying, the event sequence is kept fixed for both training and testing. Hence, given a sentence $x = w_1 \ldots w_{|x|}$, we obtain

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

To avoid word-piece segmentation, we employ a square bracket around an event type, e.g., the event token of Transport is converted to “[Transport]”. This allows us to learn more precise event token representations and yield better performance; see more discussion in Appendix A.1. Next, we apply position embeddings based on the order of event tokens following the standard setup of BERT, although linguistically, there should be no sequential difference to event types.

After that, we learn the hidden representations:

$$h_{[CLS]}, h_1^w, \ldots, h_{|x|}^w, h_{[SEP]}, h_1^e, \ldots, h_n^e = \text{BERT}($$Input),  (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 $e_{s_i}$.

**Multi-label Classifier** After learning the contextualized representations of the Input, we turn to construct the multi-label classifier. Traditional methods usually apply a Multi-Layer Perception (MLP) on the [CLS] token to yield the classifier. Differently, we feed all hidden states of event types to a MLP for the classification due to the supportive evaluation in Appendix A.2. Hence, given an input sentence $x$, we compute the predicted probability for the corresponding events by

$$\hat{p} = \sigma (\text{MLP} (h_1^e, \ldots, h_n^e)).$$  (7)

Since $\hat{p}$ is normalized to the range of 0 and 1, for simplicity, we follow (Liu et al., 2019) to determine the event labels when $\hat{p} \geq 0.5$.

Our model can then be trained by minimizing the following loss:

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

where $p_{ij} = 1$ represents the corresponding event for the $i$-th input text. Different from (Liu et al., 2019) that converts the multi-label classification task into a series of binary classification tasks, our proposal can outputs all event type(s) simultaneously.
EDM The event derangement module is the key to resolve the imbalanced learning issue. Derangement is a term in combinatorics, where a permutation of the elements in a set makes no element appearance in its original position. In the implementation, when the target (i.e., the ground-truth) events are major events, we deliver the derangement procedure with probability $q$; see line 6 of Algo. 1. Moreover, only events in $E_D$ and not pre-selected are selected from derangement; see line 9 of Algo. 1. It is noted that $E_D$ is $r$ events excluding the target events in $E_{GT}$ from the beginning of $S_{SA}$ (i.e., usually the deranged events are major events; see the definition in line 2 of Algo. 1). The parameter $r$ allows us to determine the number of the events for derangement.

The underlying effect of derangement is to compress the learning of major events, which is close to under-sampling the training instances from major events. This can yield a more balanced training process and resolve the imbalanced learning issue. We provide more supporting results and explanations in Sec. 5.2.

4 Experiments

We present the experimental setups and overall performance in the following.

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-uncased from Hugging Face (Wolf et al., 2019) is adopted as the backbone model. 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) to set $\alpha$ as 0.5 and round $k$ to the nearest integer based on the calculation by Eq. (4). In EDM, the derangement probability $q$ is set to 0.2 and $r$ is 24 from empirical selection. The batch size is 8. The dropout rate is 0.1. ADAM is the optimizer (Kingma and Ba, 2015) with a learning rate of $2 \times 10^{-5}$. We train our models for 10 epochs to give the best performance. All experiments are conducted on an NVIDIA A100 GPU in around 1.5 hours.

4.2 Overall Performance

We compare our proposed BERT_RC and BERT_DRC 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

1https://www.dropbox.com/s/4h4p0d126jha7q6/DRC.zip?dl=0
Table 1: Event detection results on both the event subtypes and event main types of the ACE2005 corpus.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Subtypes (%)</th>
<th>Main (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>R</td>
</tr>
<tr>
<td>TBNNAM (Liu et al., 2019)</td>
<td>76.2</td>
<td>64.5</td>
</tr>
<tr>
<td>DYGIE++, BERT + LSTM (Wadden et al., 2019)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DYGIE++, BERT Finetune (Wadden et al., 2019)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BERT_RC_Trigger (Du and Cardie, 2020)</td>
<td>71.7</td>
<td>73.7</td>
</tr>
<tr>
<td>DMBERT (Wang et al., 2019)</td>
<td>77.6</td>
<td>71.8</td>
</tr>
<tr>
<td>RCEE_ER (Liu et al., 2020)</td>
<td>75.6</td>
<td>74.2</td>
</tr>
<tr>
<td>DMBERT + Boot (Wang et al., 2019)</td>
<td>77.9</td>
<td>72.5</td>
</tr>
<tr>
<td>BERT Finetune</td>
<td>72.8</td>
<td>68.7</td>
</tr>
<tr>
<td>Our BERT_RC</td>
<td>76.9</td>
<td>72.3</td>
</tr>
<tr>
<td>Our BERT_DRC</td>
<td>79.5</td>
<td>76.8</td>
</tr>
</tbody>
</table>

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_RC does not access to the triggers, it attains significant better performance than TBNNAM, DYGIE++, and BERT_RC_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 RC framework is effective to learn the semantic information between given texts and event types. (2) After introducing the derangement mechanism, our proposed BERT_DRC can significantly outperform all compared methods in all three metrics. Especially, it attains 3.0% more F1 score than the best baseline, DMBERT Boot. The results show that our proposed RC framework is order-sensitive. This can be verified by the results of BERT_RC and BERT_RC_Shuffle_Test. Here, BERT_RC_Shuffle_Test is trained with the same event order of BERT_RC, but tested with a shuffled event order. By confusing BERT_RC with a different event order during inference, we obtain a significant drop on the F1 score to 12.2%. This detection: DYGIE++ (Wadden et al., 2019): a BERT-based framework modeling text spans; BERT_RC_Trigger (Du and Cardie, 2020) and RCEE_ER (Liu et al., 2020): both BERT-based models converting event extraction as an MRC 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 2: Evaluation results for event orders.

<table>
<thead>
<tr>
<th>Methods</th>
<th>P</th>
<th>R</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT_RC_Same</td>
<td>75.7</td>
<td>71.6</td>
<td>73.6</td>
</tr>
<tr>
<td>BERT_RC</td>
<td>76.9</td>
<td>72.3</td>
<td>74.7</td>
</tr>
<tr>
<td>BERT_RC_Shuffle_Test</td>
<td>18.2</td>
<td>9.2</td>
<td>12.2</td>
</tr>
<tr>
<td>BERT_DRC_Shuffle_Test</td>
<td>66.0</td>
<td>45.1</td>
<td>53.6</td>
</tr>
<tr>
<td>BERT_DRC</td>
<td>79.5</td>
<td>76.8</td>
<td>78.1</td>
</tr>
</tbody>
</table>

Table 2 reports the effect of the event orders in different cases. The first two rows record the results of BERT_RC_Same and BERT_RC, where BERT_RC_Same applies the same position embedding to all event types to eliminate the difference in event orders. On the contrary, BERT_RC applies varied position embedding to each event type. The results show that by leveraging the event order, BERT_RC gains around 1% improvement on the F1 score.

5 More Analysis

We try to discover the underlying mechanism of our proposed derangement and provide more analysis on our proposal.

5.1 Effect of Event Orders

Table 2 reports the effect of the event orders in different cases. The first two rows record the results of BERT_RC_Same and BERT_RC, where BERT_RC_Same applies the same position embedding to all event types to eliminate the difference in event orders. On the contrary, BERT_RC applies varied position embedding to each event type. The results show that by leveraging the event order, BERT_RC gains around 1% improvement on the F1 score.

We further show that our BERT_RC is ordersensitive. This can be verified by the results of BERT_RC and BERT_RC_Shuffle_Test. Here, BERT_RC_Shuffle_Test is trained with the same event order of BERT_RC, but tested with a shuffled event order. By confusing BERT_RC with a different event order during inference, we obtain a significant drop on the F1 score to 12.2%.
5.2 Effect of EDM

We show the effect of EDM to understand the underlying mechanism of how EDM works. Fig. 3(a) shows the losses of BERT_DRC and BERT_RC on major events and minor events, respectively. To amplify the effect, we set \( q = 1.0 \), an extreme case of EDM, where the event derangement is conducted on each training batch. It shows that due to the interference of the derangement, the loss of BERT_DRC on the major events drops much faster and is much smaller (close to zero) than the counterpart of BERT_RC. We conjecture that the swift convergence of BERT_DRC on major events comes from the leak of the position hint implied by derangement, because our model is order-sensitive. During derangement, the position of ground-truth (major) events is reserved while other events are deranged. Hence, this yields low loss and less gradient update on major events in BERT_DRC than that in BERT_RC. In other words, derangement prohibits major events from being excessively learned. The derangement procedure implicitly implements under-sampling the instances of major classes during training and thus fulfills a more balanced learning process. 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 (i.e. minor events in our case), which, for instance, makes the blind have stronger hearing.

Figure 3(b) further reports the performance of each event by setting the derangement probability \( q \) to 0.2 and the size of derangement set \( r \) to 24, which achieves the best performance of BERT_DRC. Via derangement or a more balanced training, BERT_DRC attains an F1 score of 78.1%, a 3.4% improvement over BERT_RC. By examining the details, the main improvement comes from recognizing the minor events, i.e., improving the F1 score from 69.1% to 72.4%.

Figure 4: Effect of \( q \) and \( r \) when evaluated on ACE2005.
5.3 Effect of Hyperparameters

In this section, we test the derangement probability $q$ and the size of the derangement set $r$, where $q$ is selected from $\{0.1, 0.2, 0.4, 0.5, 0.7\}$ and $r$ is selected 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 detecting major events. Figure 4 shows 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 than a larger one while $r$ is selected in the range of 15 and 25 because $r$ can determine the scale of perturbation. A smaller $r$ may cause negligible perturbation and a larger $r$ may affect the disturbance of the minor events. Though usually, these two parameters are data-oriented, we observe similar trend for the TAC-KBP corpus; see more results in Appendix A.5.

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

5.4 Gradient Explanation

In the literature, the gradient explanation has been verified as a more stable method to explain the attention 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 input text 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

- For the event of “Die”, our BERT_DRC 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_DRC attains low gradients on the triggers or gets high gradients on unrelated tokens, such as “to” and “.”.

- More importantly, our BERT_DRC 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 show the potential power of our proposal in not only linking triggers to the corresponding events, but also highlighting the corresponding event arguments. Our proposal can be a better tool to signify these words than traditional trigger-based methods.

6 Conclusion and Future Work

In this paper, we propose a novel Derangement Question Answering (DRC) framework on top of BERT to detect events without triggers and under the imbalanced setting. By treating the input text as a Context 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, we propose a simple yet effective derangement mechanism to relieve the imbalanced training. By delivering perturbation when the target event is a major event, we train prohibit the training and implicitly under-sample the training instance. We conduct sufficient evaluation and show that our proposed 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.

Several interesting directions can be considered in the future. First, since our proposal is event-order-sensitive, it is worthy to explore how to generate an optimal initial event order. Second, the gradient explanation is effective to signify triggers and arguments. It is meaningful to merge it in our proposal to extract the key information of events. Third, it would be worthwhile to adapt our proposal to other information extraction tasks to extend its application scope.

A Appendix

We provide more analysis to support our proposal.
Table 3: Results of different conversion ways of event tokens.

<table>
<thead>
<tr>
<th>Conversion</th>
<th>P</th>
<th>R</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>75.2</td>
<td>67.6</td>
<td>71.7</td>
</tr>
<tr>
<td>New</td>
<td>77.3</td>
<td>68.2</td>
<td>72.5</td>
</tr>
</tbody>
</table>

A.1 Effect of Event Tokens Conversion

There are two intuitive ways to treat the event tokens in our proposed 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 DRC 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 3 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.

<table>
<thead>
<tr>
<th>Input</th>
<th>P</th>
<th>R</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>[CLS]</td>
<td>77.3</td>
<td>68.2</td>
<td>72.5</td>
</tr>
<tr>
<td>All event tokens</td>
<td>76.9</td>
<td>72.3</td>
<td>74.7</td>
</tr>
</tbody>
</table>

Table 4 reports the performance of different inputs for the multi-label classifier and shows that by feeding the event representations as the input, our BERT_RC 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_DRC.

Table 5: The performance of our BERT_RC and BERT_DRC on a more balanced dataset.

<table>
<thead>
<tr>
<th>Model</th>
<th>P</th>
<th>R</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT_RC</td>
<td>76.4</td>
<td>77.8</td>
<td>77.1</td>
</tr>
<tr>
<td>BERT_DRC</td>
<td>75.0</td>
<td>76.3</td>
<td>75.6</td>
</tr>
</tbody>
</table>

Table 5 reports the comparison results of BERT_RC and BERT_DRC and shows that BERT_RC attains satisfactory results and beats BERT_DRC 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_RC 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_DRC detects more event types than the ground truth. The most event type that BERT_DRC 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_DRC 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_DRC 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_DRC 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 RCsr ... 3 words omitted... secure key oil fields.

  BERT_DRC 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_DRC 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 nonevent sentences. When there are no event types in a sentence, BERT_DRC may fail to classify it as the type of negative. This is because there is no sufficient clues for BERT_DRC to learn the patterns from the type of negative. BERT_DRC also turns out to give low predicted probabilities on all event types.

<table>
<thead>
<tr>
<th>Methods</th>
<th>P</th>
<th>R</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSEP-EMD (Peng et al., 2016)</td>
<td>69.2</td>
<td>47.8</td>
<td>56.6</td>
</tr>
<tr>
<td>BERT Finetune</td>
<td>84.1</td>
<td>65.0</td>
<td>71.7</td>
</tr>
<tr>
<td>Our BERT_RC</td>
<td>77.4</td>
<td>74.8</td>
<td>76.1</td>
</tr>
<tr>
<td>Our BERT_DRC</td>
<td>79.8</td>
<td>75.2</td>
<td>77.4</td>
</tr>
</tbody>
</table>

Table 6: The results of our BERT_RC and BERT_DRC when evaluated on TAC-KBP 2015.

![Figure 7: The event type distribution in TAC-KBP.](image)

A.5 Evaluation on TAC-KBP

We evaluate our BERT_RC and BERT_DRC on the TAC-KBP-2015 corpus (Ellis et al., 2015). The corpus is annotated with event nuggets that fall into 38 types. We process the data following (Peng et al., 2016). The data distribution is shown in Fig. 7 and is more balanced than ACE2005 with IR ≈ 61.5. We implement our BERT_RC and BERT_DRC by setting the configuration in align with that on the ACE2005 dataset (see Section 4.1). The results are shown in Table 6. It can be observed that our BERT_RC can greatly outperform the finetuned base BERT by 4.4% in F1 score. Our BERT_RC is thus a framework proven capable in both the ACE2005 English dataset and the TAC-KBP-2015 dataset. The EDM can further improve our BERT_DRC by 1.3%. But the increment is relatively smaller than that on the ACE2005 dataset, which may be caused by the different corpus and data distribution of two datasets.

Based on the TAC-KBP-2015 dataset, we investigate the effect of q and r in EDM on BERT_DRC’s
types in practice.

performance. We select $q$ from $\{0.1, 0.2, 0.3\}$ and $r$ from $\{3, 6, \ldots, 33\}$.

Figure 9 shows the performance with respect to $r$ for different $q$. It is shown that the best performance is attained when $q = 0.2$ and $r = 21$, reaching 77.8% for F1 score. The trends remain largely the same as those on the ACE2005 dataset. The best performances also occur when $r$ 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.

We then conduct gradient explanation on our DRC framework as in Sec. 5.4. We randomly choose instances from the test set and visualize gradients respect to the correctly predicted event types by our BERT_DRC. As shown in Fig. 8, “nominated”, “leaked” and “sentences” are respectively triggers for those three sentences and receive significant positive gradients compared with other words. This shows that our DRC framework can automatically learn to spot triggers and relate them to event