DPNPED: Dynamic Perception Network for Polysemous Event Trigger Detection

JUN XU AND MENGSHU SUN
Ant Financial Services Group, Hangzhou, Zhejiang 310058, China
Corresponding author: Jun Xu (xj169860@antfin.com)
This work was supported by the Ant Financial Services Group through Ant Innovative Research Program.

ABSTRACT Event detection is the process of analyzing event streams to detect the occurrences of events and categorize them. General methods for solving this problem are to identify and classify event triggers. Most previous works focused on improving the recognition and classification networks which neglected the representation of polysemous event triggers. Polysemy is habitually somewhat confusing in semantic understanding and hard to detect. To improve polysemous trigger detection, this paper proposes a novel framework called DPNPED, which dynamically adjusts the network depth between polysemous and common words. Firstly, to measure the polysemy, the difficulty factor is devised based on the frequency of a word as an event trigger. Secondly, the DPNPED utilizes a confidence measure to automatically adjust the network depth by comparing the predicted and initial probability distribution. Finally, our model applies focal loss to dynamically integrate the difficulty factor and confidence measure to enhance the learning of polysemous triggers. The experimental results show that our method achieves a noticeable improvement in polysemous event trigger detection.

INDEX TERMS Artificial intelligence, text mining, text recognition.

I. INTRODUCTION Event detection (ED) is an aspect of information extraction, that aims at detecting event triggers in text and classifying them into specific event types [1], [2], [3], [4], [5], [6]. In general, the event trigger is a keyword or phrase to describe events. For instance, as the intuitive sentence shown in Table 1, the ED system is required to identify two events: an Attack event triggered by “hacked”, and a Die event triggered by “death”. It is easy to see that some triggers are simple to detect, but others are difficult to identify. Take the benchmark ACE2005 as an example, as shown in Table 1, judging the Attack event triggered by “hacked” and the Die event triggered by “death” is straightforward. On the contrary, it is confusing to determine whether “going” is a Transport event, which means Obama goes to the place; or a Start-Position event, which means Obama will work in the White House.

| Intuitive Sentence | Polysemous Sentence |
|--------------------|---------------------|
| At least three members of a family in India’s northeastern state of Tripura were hacked to death by a tribal mob for allegedly practicing witchcraft, police said Thursday. | Obama is going to the White House. Polysemous Trigger: going |
| Intuitive Event: Attack | Polysemous Event: Transport |
| Intuitive Event: Die | Polysemous Event: Start-Position |

From the results of the baseline model, it can be analyzed that the event triggers which are difficult to detect are mainly concentrated in polysemous words, such as, “go”, “force” and “take”. However, most of the previous works [5], [7], [8], [9], [10] focused on improving the recognition and classifi-
cution networks, neglecting the representation of polysemous event triggers. Therefore, these methods are not effective in detecting polysemous triggers.

Based on previous experience, for polysemous triggers, we need more context to decrease ambiguity. To get more contextual information, it is usually to use external knowledge [9], [11] or increase the network depth [12], [13]. For reusability, we are inclined to adjust the network depth. Regarding the dynamic semantic representation of words, [14] applied adaptive computation time for recurrent neural networks, which could dynamically increase the number of network layers. Reference [13] designed a universal transformer for BERT based on adaptive computation time, which was improved on a variety of challenging questions, such as language modeling, reasoning question, and answer, etc. Overall, the core issue of dynamic networks is how to determine whether the network needs to stack more layers. In [14] and [15], the sigmoid probability of RNN cells or transformers is accumulated until it exceeds a given threshold. The problem with these methods is that the cumulative sigmoid probability as a halting condition is task-irrelevant and unexplainable. This halting condition means that the representation of all tasks is similar. To provide a task-oriented and reasonable halting mechanism, we design a confidence measure that is derived from the divergence [15] calculated by the contrast between two probability distributions. The default probability distribution is uniform initialization. In each layer, we calculate the target probability distribution. Compare the two distributions, if the difference is greater than the given confidence measure score, then stop increasing layers. Otherwise, we should stack more layers.

To improve the semantic representation of polysemous triggers, this paper applies a difficulty factor and confidence measure to automatically adjust the network depth between polysemous and common words. As shown in Table 2, the word “go” appears 271 times in the training set, 44 of which are triggers, and the frequency of it as an event trigger is 0.162. At the same time, “go” acts as a trigger for various event types, such as, “Transport” and “End-Organization”. Obviously, compared with the trigger “kill”, the trigger “go” is more difficult to detect. To make our model focus on polysemous triggers, we define a difficulty factor to measure the polysemous of a word. The difficulty factor is used for the focal loss to enhance the learning of polysemous triggers.

We have conducted extensive experiments on ACE 2005 and MAVEN datasets. The experimental results show that our model achieves specific improvement on polysemous event trigger detection.

### II. METHODOLOGY

In this paper, we formulate event trigger detection as a sequence labeling task. Formally, given a sentence \( X = [x_1, x_2, \ldots, x_n] \) of \( n \) words and an index \( i \) of the trigger candidate \( x_i \), to predict the event type \( y \) of this candidate. We define a start and end tag for each word. The tag “O” stands for “other” event, which means that the corresponding word is not an event trigger. The tag “EventType” represents it is a trigger. For example, as can be seen in Figure 1, the trigger word “hacked” is annotated as “Attack” [5], [16], and our model marks all words in the sentence sequentially via the dynamic perception network, where “hacked” has a deeper network than “death”.

#### A. TRIGGER ENCODER

This paper takes token embedding as input. In the embedding layer, each token \( x_i \) consists of the following vectors:

- The word embedding vector of each token: this is obtained by looking up a pre-trained model [17]. The vector can be represented as \( w_i \), and it contains token embedding, segment embedding, position embedding.
- The candidate embedding vector of each token: it can be obtained from the training data, if a word appears as a trigger word, set the candidate to 1. Otherwise, set it to 0. The vector can be represented as \( t_i \).
- The pos embedding vector of each token: the part-of-speech tagging of each word can be obtained from the stanza tool. A total of 57 tags are used in this paper. The vector can be represented as \( p_i \).

The embedding of each token can be formulated as \( x_i \), which is fed into the dynamic perception network:

\[
x_i = w_i + t_i + p_i
\]

Specifically, as shown in Figure 2, taking \( x_i \) as the input. It will be transformed by the embedding layers to a vector

| Word | Freq | Type | Polysem | Difficulty |
|------|------|------|---------|------------|
| go   | 44/271 = 0.162 | 8 | high | 0.9979 |
| take | 23/198 = 0.116 | 6 | high | 0.9998 |
| meet | 39/75 = 0.520 | 3 | low | 0.8725 |
| fight | 83/129 = 0.643 | 4 | low | 0.8257 |
| war  | 338/379 = 0.851 | 2 | low | 0.7641 |
| kill | 138/141 = 0.979 | 1 | low | 0.7353 |
**FIGURE 1.** Framework overview of dynamic perception network for event detection. The dynamic perception network adopts the difficulty factor and confidence measure to adjust the network depth between polysemous and common words. It can be seen that, compared to other words, ‘hacked’ has a deeper network.

**FIGURE 2.** The basic working principle of dynamic perception network. CS represents the number of computational steps, also known as the number of network layers. In each layer, the predicted probability distribution is calculated, and then compare it with the initial probability distribution to determine whether the model needs to stack more layers.

representation $e_i$ as follows:

$$e_i = \text{Embedding}(x_i)$$  \hspace{1cm} (2)

Next, the transformer blocks in the encoder performs a layer-by-layer feature extraction as equation 3:

$$h_i^{cs} = \text{Transformer}(h_i^{cs-1})$$  \hspace{1cm} (3)

where CS is the number of computational steps. The predicted probability distribution of the current hidden state like equation 4:

$$p_i^{cs} = \text{softmax}(Wh_i^{cs} + b)$$  \hspace{1cm} (4)

**FIGURE 3.** The confidence measure score of different predictions. $q_j^i$ is a uniform distribution which is the red dotted line. $p_j^i$ is the predicted probabilities of our dynamic network. The confidence measure score can be calculated by equation 6. The more the predicted probability distribution deviates from the uniform distribution, the larger the confidence measure score.

**B. CONFIDENCE MEASURE**

In order to measure whether this layer has enough information to predict the target correctly, we design a confidence measure to estimate the disparity between the predicted probability distribution and the initial probability distribution. Formally, the confidence measure score of the $i$-th token in $j$-th layer is expressed as follows:

$$c^i_j = \frac{1}{\log M} \sum_{k=1}^{M} p_j^i(k) \log \frac{p_j^i(k)}{q_j^i(k)}$$  \hspace{1cm} (5)

$$q_j^i(k) = \frac{1}{M}$$  \hspace{1cm} (6)

where $M$ is the number of labeled classes, and $p_j^i(k)$ is the probability of the $i$-th token in $j$-th layer belongs to label $k$, and $p_j^i \in \mathbb{R}^M$. The factor $\frac{1}{\log M}$ is used to balance the discrepancy in different tasks. As shown in Figure 3, the more
the predicted probability distribution deviates from the initial distribution, the larger the confidence measure score.

If the confidence measure score $c_i$ of the input $x_i$ is greater than the given threshold value $\mu$ (a constant), the bounding step is located. Otherwise, we need to stack more layers and the computational step $CS_i$ of the input $x_i$ should be increased:

$$CS_i = CS_i + 1 \quad \text{(7)}$$

To avoid making mistakes, the dynamic perception network tends to learn as long as possible. Therefore, we set the maximum number of computational steps ($MC$). If the computational step $CS_i \geq MC$, then $CS_i$ is the boundary of input $x_i$, and the output of this input is the last layer.

C. DIFFICULTY FACTOR

We could notice that some triggers are easy to detect, but others are baffling to identify. To make our model focus on polysemous triggers, we define a difficulty factor to measure the polysemy of a word. Practically, the difficulty factor is calculated by the frequency of a word as an event trigger. The prior probability is expressed as follows:

$$f_i = \frac{\text{count}(\text{triggers})}{\text{count}(\text{words})} \quad \text{(8)}$$

where $\text{count}(\text{triggers})$ represents the number of a word as an event trigger and $\text{count}(\text{words})$ means the occurrences of this word in the training corpus. For example, as can be seen from Table 2, the word ‘war’ appears 379 times in the training set, 338 of which are triggers, and the frequency of it as an event trigger is 0.1048. Table 2, the word ‘war’ appears 379 times in the training set, 338 of which are triggers, and the frequency of it as an event trigger is 0.1048. For more details, please refer to Appendix B.

D. DYNAMIC FUSION LOSS

To learn the difference of each word, during training, we use the focal loss that adds a factor $d_i \times (1 - p_i(k))^\gamma$ to the standard cross-entropy criterion. Setting $\gamma > 0$ reduces the relative loss for well-classified examples, putting more focus on hard, misclassified triggers. The total loss of our model $L(x)$ in the sequence $x$ is formulated as:

$$L(x) = \alpha \sum_{i=1}^{N} \frac{CS_i}{MC} - \sum_{i=1}^{N} \sum_{k=1}^{M} d_i (1 - p_i(k))^\gamma \log p_i(k) \quad \text{(10)}$$

where $N$ is the input sequence length, and $M$ is the number of labeled classes. $d_i$ is the difficulty factor of the input $x_i$, and $p_i(k)$ is the probability of the $i$-th token belongs to label $k$. As $CS_i \gg$ focal loss, we divide $CS_i$ by $MC$, and $\alpha$ is a balance factor.

E. SPAN PREDICTION

Given the output of DPN, the model first predicts the probability of each token being a start index as follows:

$$p_{start} = \text{softmax}(W_{start}h^{cs} + b) \quad \text{(11)}$$

where $h^{cs} \in \mathbb{R}^{N \times d}$ is the output of DPN, and $N$ is the sequence length; $d$ is the hidden dimension. The start position predicted probability is $p_{start} \in \mathbb{R}^{N \times M}$, and $M$ is the number of classes. The end index prediction procedure is exactly the same, except that we have another matrix $W_{end}$ to obtain probability matrix $p_{end} \in \mathbb{R}^{N \times M}$. Finally, our model starts with the $\text{start-index prediction}$, and from the $\text{end-index prediction}$ finds a nearest position of the same event type. At training time, the input $X$ is paired with two label sequences $y_{start}$ and $y_{end}$ of length $N$ representing the ground-truth label of each token $x_i$ being the start index or end index of any trigger. Therefore, we have the following two losses for start and end index predictions:

$$L_{start} = L(p_{start}, y_{start})$$
$$L_{end} = L(p_{end}, y_{end}) \quad \text{(12)}$$

where the loss function $L$ is equation 10. The overall training objective to be minimized is as follows:

$$L = L_{start} + L_{end} \quad \text{(13)}$$

III. EXPERIMENTAL SETTINGS

A. Datasets

We conduct experiments on the widely used ACE2005 and MAVEN dataset [10]. For more statistical details, please refer to Appendix A.

B. Evaluation Metric

The evaluation metric is similar to previous works [10], [18]. For more details, please refer to Appendix B.

C. Implementation Details

During the training, we use the same embedding as BERT [17] for input sequence initialization. The token, segmentation, and position embeddings are initialized from the pretrained model. The candidate and pos embedding use random initialization. The implementation of the transformer block is identical with Vaswani [19]. After a transformer block learning, the confidence measure score between the output probability distribution and the uniform distribution is calculated by equation 6. If the confidence measure score $c_i$ of the input $x_i$ is greater than the given threshold value $\mu$, the output of the current transformer block is the final output and the final transformer block will be copied until the number of layers accumulates to MC. For other hyper-parameters and details, please refer to Appendix C.

D. Comparison Methods

For comparison, we investigate the performance of the following state-of-the-art methods: (1) MOGANED [20],
TABLE 3. The overall trigger detection performance of various models on ACE 2005 and MAVEN. The evaluation metrics are micro precision, recall, and F1 score (%). Bold marks the highest number among all models, underline marks the second-highest number.

| Models     | ACE 2005 | MAVEN |
|------------|----------|-------|
|            | P        | R     | F1   | P    | R     | F1   |
| MOGANED    | 72.5     | 72.3  | 75.7 | 63.4 | 64.1  | 63.8 |
| DMBERT     | 77.9     | 72.5  | 75.1 | 62.7 | **72.3** | **67.1** |
| GatedGCN   | 78.8     | 76.3  | 77.6 | 65.4 | 67.6  | 66.5 |
| EE-GCN     | 76.7     | 78.6  | 77.6 | 64.2 | 68.8  | 66.4 |
| MLBiNet    | 74.7     | **83.0** | 78.6 | 61.7 | 70.7  | 65.9 |
| **DPNPED** | **81.2** | **77.3** | **79.3** | **70.3** | **68.6** | **69.4** |

TABLE 4. Statistics of the ACE2005 test set and MAVEN dev set. #High means the number of highly polysemous triggers and #Low represents the number of common triggers. The ratio stands for the proportion of highly polysemous triggers.

| Dataset       | #Low | #High | #All | Ratio |
|---------------|------|-------|------|-------|
| ACE2005 test set | 306  | 127   | 433  | 29.3% |
| MAVEN dev set  | 15.145 | 4.413 | 19.558 | 22.6% |

which uses graph convolution network with aggregative attention to explicitly model and aggregate multi-order syntactic representations. (2) DMBERT [7], which augments the training data with external unlabeled data by adversarial mechanism. (3) GatedGCN [18], which presents a gating mechanism to filter noisy information in the hidden vectors of the GCN models for event detection based on the information from the trigger candidate. (4) EE-GCN [21], which exploits syntactic structure and typed dependency label information to perform event detection. (5) MLBiNet [22], which applies a multi-layer bidirectional network to capture the document-level association of events and semantic information simultaneously. For more implementation details and hyper-parameters, please refer to Appendix D.

IV. EXPERIMENTAL RESULTS

A. OVERALL RESULTS

As can be seen from Table 3, our model is leading the way on both datasets. At the same time, this model won 1st place in the MAVEN competition. From the results, we have the following observations: (1) Compared to the best method (MLBiNet) in ACE2005, our F1 score is relatively improved by 0.8%; (DMBERT) in MAVEN, our F1 score is relatively promoted by 3.4%. (2) MLBiNet, EE-GCN and GatedGCN perform well on ACE 2005, but these models perform poorly in MAVEN. In MAVEN, compared with MLBiNet, the F1 score of our model is relatively improved by 5.3%. This result shows that our model has better generalization performance.

B. RESULTS FOR POLYSEMOUS TRIGGERS

To evaluate the effect of our method on extracting polysemous triggers, we divide the ACE2005 test set and MAVEN dev set (the MAVEN test set is a blind set, which only supports online testing) into two parts (low and high) and perform evaluations separately. Among all event triggers, if the probability of a word as an event trigger is less than 0.5, it is a highly polysemous trigger; otherwise, it is a common trigger. As shown in Table 4, more than 20% of triggers in ACE and MAVEN are highly polysemous triggers.

As can be seen in Table 5, benefiting from the dynamic perception network and difficulty factor, compare with MLBiNet and DMBERT, the F1 score of our model improves significantly in highly polysemous triggers which increased by 8.8% in the ACE2005 test set and 9.5% in the MAVEN dev set. In ACE2005, MLBiNet is 1.9% higher than our model in common trigger (Low) recognition, and the main reason is that they use the document-level association of events to solve the multi-event problem. However, the multi-event problem in MAVEN is not obvious, the effect of MLBiNet deteriorates drastically, and our model is still robust.

C. ABSTRACTION ANALYSIS

We ablate each part of our model on the ACE 05 and MAVEN, as shown in Table 11. First, without difficulty factor, we observe performance drops of 1.54% on ACE2005 and 0.73% on MAVEN, which verifies the usefulness of difficulty factor in polysemous triggers extraction. By removing pos embedding, and trigger candidate embedding, the performance drops slightly. Furthermore, after replacing DPN with LSTM or BERT, the performance decrease is most significant. The main reason is that LSTM and BERT are difficult to extract polysemous trigger, resulting in boundary errors. This shows that the DPN module increases the computational
TABLE 7. The F1 score (%) of highly polysemous triggers on the ACE2005 test set and MAVEN dev set. "Freq" means the proportion of a word as an event trigger. "CS" represents the average number of computational steps. Δ stands for relative change.

| Dataset | Word | Freq | CS | BERT | DPN | Δ    |
|---------|------|------|----|------|-----|------|
| MAVEN   | fair | 0.152| 14 | 54.5 | 63.6| 16.7%|
|         | show | 0.239| 17 | 58.8 | 66.7| 13.4%|
|         | war  | 0.309| 13 | 63.3 | 69.3| 9.5% |
|         | threat| 0.228| 19 | 60.0 | 80.0| 33.3%|
|         | fire | 0.277| 14 | 58.6 | 65.5| 11.8%|
|         | draw | 0.262| 17 | 63.6 | 72.7| 14.3%|
|         | force| 0.088| 22 | 54.6 | 62.7| 14.8%|
|         | avg  | 0.293| 16 | 58.0 | 63.6| 9.7% |
| ACE     | avg  | 0.254| 17 | 64.8 | 75.5| 16.5%|

TABLE 8. Parameter size, inference time (ms/batch), average layers for words between BERT and DPNPED model.

| Models      | params | ACE 2005 layers | ACE 2005 infer | MAVEN layers | MAVEN infer |
|-------------|--------|-----------------|---------------|--------------|------------|
| BERT<sub>base</sub> | 121M   | 12              | 212           | 12           | 270        |
| MC = 12     | 82M    | 7               | 179           | 8            | 223        |
| decline     |        | 32%             | 42%           | 16%          | 33%        |
| BERT<sub>large</sub> | 368M   | 24              | 475           | 24           | 533        |
| MC = 24     | 135M   | 11              | 382           | 13           | 411        |
| decline     |        | 63%             | 54%           | 20%          | 46%        |

As shown in Table 9, in ACE2005 and MAVEN, compared to BERT large, although the number of layers of the DPNPED model is reduced by 54% and 46%, the F1 score is increased by 6.3% and 1.8%. According to previous works [23], [24], as the number of BERT layers increases for some words, their semantics will gradually converge, which is called the semantic overfitting of words. Although each word has 24 layers in BERT large, it leads to the semantic overfitting of some words. However, our dynamic perception network reduces unnecessary layers through the confidence measure, thus reducing the semantic overfitting problem of words. Compared with MC = 24, the effect of MC = 12 is 0.6% worse in ACE2005 and 0.4% less in MAVEN. The maximum number of computational steps limits the learning ability of the model.

TABLE 9. Performance comparison of different methods on the test set with gold-standard triggers.

| Models          | ACE 2005 | MAVEN |
|-----------------|----------|-------|
|                 | P   | R   | F1  | P   | R   | F1  |
| BERT<sub>base</sub> | 72.2 | 76.9 | 74.5 | 65.5 | 70.8 | 68.0 |
| MC = 12         | 80.8 | 76.9 | 78.8 | 70.0 | 68.2 | 69.1 |
| BERT<sub>large</sub> | 72.5 | 76.8 | 74.6 | 65.8 | 70.7 | 68.2 |
| MC = 24         | 81.2 | 77.3 | 79.3 | 70.3 | 68.6 | 69.4 |

TABLE 10. The F1 score (%) of different difficulty factors on the ACE2005 test set and MAVEN dev set. Bold marks the highest number among all models, underline marks the second-highest number.

| Models          | ACE 2005 | MAVEN |
|-----------------|----------|-------|
|                 | Low | High | All | Low | High | All |
| d<sub>i</sub> = d<sub>i</sub><sub>max</sub> | 80.9 | 75.5 | 79.3 | 72.3 | 63.6 | 70.3 |
| d<sub>i</sub> = 1 | 78.8 | 69.9 | 76.2 | 71.1 | 59.7 | 68.5 |
| d<sub>i</sub> = 0.731 | 80.1 | 72.6 | 77.9 | 70.9 | 61.6 | 68.8 |
| d<sub>i</sub> = 1 | 80.2 | 73.0 | 78.1 | 71.1 | 64.1 | 68.9 |

D. ANALYSIS OF DYNAMIC PERCEPTION NETWORK

The dynamic perception network employs a confidence measure to adjust the depth of neural networks. As can be seen in Table 7, we listed a few highly polysemous triggers. For example, the frequency of “threat” as an event trigger is 0.228. Compared with BERT+Span, DPN+Span uses 19 computational steps, and the F1 score is relatively increased by 33.3%. As the computational step increase, DPNPED will take more interaction with its context that can better capture the contextual semantics. On the whole, the overall F1 score has increased by 9.7% in MAVEN and 16.5% in ACE2005 on the highly polysemous triggers. The results on common trigger, please refer to Appendix E.

As shown in Table 8, when the maximum computational step is set to 24, our DPNPED model parameter is 135M, which is 63% smaller than BERT large. The main reason is that DPNPED uses confidence measure to reduce the number of network layers (46%-54% decline) for some semantically expressive words. As the confidence measure score is calculated for each layer, the inference time is only reduced by 17%-20%.

E. ANALYSIS OF DIFFICULTY FACTOR

How to measure the difficulty of text reading? The Lexile Framework for Reading [25] is widely accepted in the world. The framework measures are based on two factors: semantic difficulty (word frequency) and syntactic complexity (sentence length). Both of these factors, over decades of research, have been shown to be excellent predictors of how difficult a text is to comprehend. In practice, we fill all the sentences to a fixed length. Therefore, we are concentrating on the semantic difficulty. For comparison, we define five different types of semantic difficulty. There are $\lambda$ and $\sigma(1/\lambda)$ for dynamic semantic difficulty and $d_i = 0.731$, 1 as the static semantic difficulty. In order to learn hard word better, this paper applies focal loss [26], [27], [28] to standard cross entropy loss, and it will make our model focused on hard, misclassified words. In the focal loss, the difficulty factor $d_i$ dynamically adjusts the loss weight of polysemous words and common words.

As shown in Table 10, in dynamic semantic difficulty, compared with $\lambda$, the F1 score of $\sigma(1/\lambda)$ in highly polysemous triggers is relatively increased by 8.0% (ACE2005) and 6.5%
Compared with static semantic difficulty, dynamic semantic 1 means that focal loss only adjusts the balance coefficient. According to the statistics of a large number of experimental results, $\mu$ will be reduced, and our model will underfit. According to the Figure 4, if to determine whether more layers are needed. As shown in Figure 4, if $\mu$ is too small, the number of computational steps fluctuates greatly, and the sigmoid function can limit it to $[0, 0.731]$. In static semantic difficulty, $d_i = 0.731$ or $d_i = 1$ means that focal loss only adjusts the balance coefficient. Compared with static semantic difficulty, dynamic semantic difficulty $\sigma(f, \mu)$ is increased by 3.4% (ACE 2005) and 3.2% (MAVEN) in highly polysemous triggers. Dynamic semantic difficulty allows triggers that are difficult to identify in sentences to get more attention.

**F. HYPERPARAMETER SENSITIVITY ANALYSIS**

The modulating factor of focal loss $\gamma = 0$, which is equivalent to cross entropy loss. It can be seen from the experiment that focal loss is more suitable for event detection tasks with event imbalanced. The confidence measure is mainly used to determine whether more layers are needed. As shown in Figure 4, if $\mu$ is too small, the number of computational steps will be reduced, and our model will underfit. According to the statistics of a large number of experimental results, $\mu = 0.5$ is the most suitable value.

**V. LIMITATIONS**

Nonetheless, these results must be interpreted with caution and a number of limitations should be borne in mind. Firstly, although the DPN design is very versatile and suitable for all sequence labeling problems, the improvement is not obvious in the NER and part-of-speech tagging experiments. Secondly, the difficulty factor depends on the training data, so it is not suitable for the recognition of unregistered words.

**VI. RELATED WORK**

As reported by [18], [31], [32], and [33], pre-trained language model turns out to require deeper layers when long-distance dependency information is required. However, in the actual situation, the deeper model requires a lot of computational resources and data to fine-tune. Therefore, in the interests of both computational efficiency and ease of learning, it seems preferable to dynamically vary the number of layers for different inputs. Dynamic neural networks [2], [13], [14], [32] change their architectures based on the input data. Networks with dynamic depth [33], [34], [35], [36] achieve deep semantic representation in two ways, early exiting when shallower sub-networks have high classification confidence [36], [37], [38], or adding additional layers adaptively [39], [40]. For the halting conditions of dynamic networks, [41], [42], [43] proposes a contrastive learning approach to measure semantic integrity based on divergence, which can be determined whether to stop early or to add additional layers. Although dynamic neural networks can enrich the semantic information of polysemous triggers, it is still difficult to fully understand those low-frequency polysemous triggers. Therefore, it needs to pay more attention to these hard samples. For the critical evaluation of text difficulty, [44], [45], [46] publishes characterizing text difficulty with word frequencies. [26], [27] designs a focal loss to deal with hard samples; [47] uses contrastive learning by selecting hard negative samples. Various methods are proposed for event trigger detection [1], [3], [5], [6]. However, the above methods treat all the triggers equally, leading to these methods being insufficient in dealing with polysemous triggers. To learn a better representation, we design a dynamic perception network that can adjust the network depth on polysemous triggers.

**VII. CONCLUSION**

In this work, to address polysemous triggers, we propose the dynamic perception network which applies a confidence measure to determine whether more layers are needed and difficulty factor to the cross-entropy loss in order to focus learning on hard triggers. Our model achieves a certain improvement on the polysemous event trigger detection task of ACE 2005 and MAVEN datasets.

**APPENDIX A DATASET STATISTICS**

Comparison with previous works [1], [5], [10], [11], [18], [48], we adopt the same training, validation, and testing set. The overview of the ACE 2005 and MAVEN dataset is shown in Table 12.

**APPENDIX B EVALUATION METRIC**

Our evaluation metrics are identical to all the previous works for meaningful comparison. The comparison is based on the famous F measure (F1), Precision (P) and Recall (R), $P = \frac{TP}{TP+FP}$, $R = \frac{TP}{TP+FN}$, $F1 = \frac{2PR}{P+R}$. TP (True Positive) is the number of triggers assigned correctly. FP (False Positive) is the number of triggers that do not belong to these event types.

| TABLE 11. Sensitivity analysis of hyperparameter in ACE 2005 and MAVEN. |
|------------------------|--------|--------|--------|--------|--------|--------|
| Models | ACE 2005 | MAVEN |
|        | P  | R  | F1 | P  | R  | F1 |
| MC=24  | 81.2 | 77.3 | 79.3 | 70.3 | 68.6 | 69.4 |
| MC=12  | 80.8 | 76.9 | 78.8 | 70.0 | 68.2 | 69.1 |
| $\mu = 0.5$ | 81.2 | 77.3 | 79.3 | 70.3 | 68.6 | 69.4 |
| $\mu = 0.4$ | 80.3 | 76.8 | 78.5 | 69.6 | 67.6 | 68.6 |
| $\gamma = 0$ | 80.4 | 77.1 | 78.7 | 69.9 | 68.1 | 69.0 |
| $\gamma = 1$ | 80.2 | 77.8 | 79.0 | 70.0 | 68.2 | 69.1 |
| $\gamma = 2$ | 81.2 | 77.3 | 79.3 | 70.3 | 68.6 | 69.4 |

| FIGURE 4. The F1 score of different confidence measure threshold $\mu$. |

| TABLE 12. Statistics of ACE2005 and MAVEN datasets. The #Types shows the number of the event types (i.e. the ‘Attack’ of ACE and MAVEN). |
|------------------------|--------|--------|--------|--------|
| Dataset | #Tokens | #Sentences | #Types | #Events |
| ACE2005 | 303k | 4,839 | 33 | 4,090 |
| MAVEN | 1,276k | 111,611 | 168 | 29,193 |
APPENDIX C IMPLEMENTATION DETAILS

The confidence measure threshold \( \mu \) is 0.5 and the max computation step \( MC \) is 24. Our model is trained using the Adam optimizer with a learning rate of \( 2e^{-5} \) with decay rate 0.99, and the weight decay parameter is 0.01. The dropout rate is set to 0.1, the maximum sequence length is 512 and the batch size is 8. In the inference, the difficulty factor of each word is the same as that in the training which is calculated from training data. The loss function balance factor \( \alpha \) is 0.2.

APPENDIX D COMPARISON METHODS DETAILS

In this section, we provide more implementation details of baselines. For a fair comparison, all of these models are tested using the NVIDIA TESLA P100 GPU. Most of the previous models take event detection as a sequence labeling problem. They use the “B/I/O-trigger” schema to extract event triggers. MOGANED uses a dependency tree based graph convolution network with aggregative attention to explicitly model and aggregate multi-order syntactic representations in sentences. The dependency tree is built on Stanford CoreNLP toolkit and the pre-trained word embedding is glove 6B 100d. The batch size, training epoch, and the penalty parameter is 0.0. The dropout rate as 0.99, and the weight decay parameter is 0.5 and penalty coefficient as \( 2e^{-5} \). The layers of bidirectional network is 5 and the number of GCN layers is 2. Other parameters take default values. EE-GCN exploits syntactic structure and typed dependency label information to perform event detection. The framework is the same as GatedGCN, and the dependency tree is built on Stanford CoreNLP toolkit. All hyperparameters take default values. MLBiNet takes event detection as a seq2seq problem based on multi-layer bidirectional network. The dropout rate as 0.5 and penalty coefficient as \( 2e^{-5} \). The layers of bidirectional network is 2 and other hyperparameters take default values.

APPENDIX E DYNAMIC PERCEPTION NETWORK ON COMMON TRIGGER

As can be seen in Table 13, we listed a few common triggers. Overall, the average number of computational steps for the common trigger is less than that for the polysemous trigger. The advantage of DPNPED in common trigger is not obvious. The main reason is that the semantics of common triggers are relatively clear.

APPENDIX F CASE STUDY

As can be seen in Table 14, the ambiguity of the trigger ‘go’ is serious. At the same time, the computational steps of our model will vary with context. The experimental results show that the dynamic adjustment of the network depth is better than the fixed one.

APPENDIX G ERROR ANALYSIS

To analyze the bad case in event detection, we randomly sampled 50 bad cases and gave some reasonable explanations for our model predicting error. We give them a summary in Table 15.

For recall, rare triggers (such as ‘free’, ‘end’, etc.) in the training which appear less than five times. They are the main reasons for predicting error. Annotation errors are also a severe problem. For example, ‘On the news that Jessica Lynch is eventually going to come home’, ‘come’ in the sentence should represent the transport event, but it is marked as the meet event. Others include ambiguous and indirect triggers such as ‘go’ and ‘the’, which need more in-depth semantic analysis of the sentence to identify them.

For precision, rare triggers (such as ‘go’, ‘end’, etc.) in the training which appear less than five times. They are the main reasons for predicting error. Annotation errors are also a severe problem. For example, ‘The deadliest conflict since World War II and has been going on for five years’, ‘World’ should mark as B-Attack, ‘War’ should mark as I-Attack and ‘II’ should mark as I-Attack. Others include ambiguous and indirect triggers.

---

2MOGANED code is available in https://github.com/I30rus/MOGANED
3DMBERT code is available in https://github.com/Bakser/DMBERT
4GatedGCN code is available in https://github.com/lauviet/ed-gated-gcn
5EE-GCN code is available in https://github.com/cuishiyao96/ee-gcned
6MLBiNet code is available in https://github.com/zjunlp/DocED

---

| Dataset | Word | Freq | CS | BERT | DPN | \( \Delta \) |
|---------|------|------|----|------|-----|---------|
| MAVEN   | battle | 0.704 | 14 | 68.4 | 70.3 | 2.8% |
|         | halt  | 0.777 | 14 | 68.0 | 73.2 | 7.6% |
|         | fly   | 0.874 | 9  | 70.2 | 67.1 | -4.4%|
|         | appeal| 0.618 | 9  | 66.6 | 63.8 | -4.2%|
|         | pursue| 0.739 | 13 | 60.0 | 63.4 | 5.6% |
|         | set   | 0.754 | 11 | 55.7 | 63.1 | 13.3%|
|         | bomb  | 0.504 | 8  | 60.2 | 61.5 | 2.1% |
|         | ...   | ...   | ...| ...  | ...  | ...    |
|         | avg   | 0.631 | 12 | 70.9 | 72.3 | 2.0% |
| ACE     | avg   | 0.609 | 9  | 78.5 | 80.9 | 3.1% |

---

To show that the dynamic adjustment of the network depth is better than the fixed one.
TABLE 14. An example of ‘go’- acts as triggers in different events. Layers represent the number of computational steps for these words.

| Type     | Sentence                                                                 | Layers |
|----------|---------------------------------------------------------------------------|--------|
| Transport| Every Wednesday, we go down to the uh River Bank Park.                     | 7      |
| End Posi-| It is no coincidence that the employees let go included substantially all those who had offered to transfer to Crossville. | 12     |
| Die      | I think that now he is gone, his family is still around and Barb Yeah.    | 14     |
| Start Posi- | Where’s he going to Commerce?                                            | 14     |
| Attack   | All morning, the Kurds have been trying to advance, and that was the shell going off. | 12     |
| End Org  | At the same time 2 weapons go off in Washington DC and take out the President. | 15     |
| Meet     | Secretary of State Colin Powell goes to the U.N. tomorrow to do what he can. | 9      |
| Arrest Jail | You can go back to prison for violating the conditions of your parole, for a period of time up until your sentence runs out. | 18     |

TABLE 15. Error analysis for precision and recall.

| Metric     | Reasons                          | Count |
|------------|----------------------------------|-------|
| Precision  | Rare triggers                    | 32    |
|            | Annotation errors                | 12    |
|            | Others                           | 6     |
| Recall     | Rare triggers                    | 27    |
|            | Multiple triggers inherently     | 10    |
|            | Others                           | 13    |

ACKNOWLEDGMENT
The authors would like to thank the reviewers for their helpful comments and suggestions.

REFERENCES
[1] Y. Chen, H. Yang, K. Liu, J. Zhao, and Y. Jia, “Collective event detection via a hierarchical and bias tagging networks with gated multi-level attention mechanisms,” in Proc. Conf. Empirical Methods Natural Lang. Process., 2018, pp. 1267–1276.
[2] D. Saunders, F. Stahlberg, A. de Gispert, and B. Byrne, “Domain adaptive inference for neural machine translation,” in Proc. 57th Conf. Assoc. Comput. Linguistics, A. Korhonen, D. R. Traum, and L. Márquez, Eds. Florence, Italy: Association for Computational Linguistics, 2019, pp. 222–228, doi: 10.18653/v1/p19-1022.
[3] M. Aldawarsi and M. Finlayson, “Detecting subevents using discourse and narrative features,” in Proc. 57th Annu. Meeting Assoc. Comput. Linguistics, 2019, pp. 4780–4790.
[4] T. Zhang, L. Zhang, W. Ye, B. Li, J. Sun, X. Zhu, W. Zhao, and S. Zhang, “Point, disambiguate and copy: Incorporating bilingual dictionaries for neural machine translation,” in Proc. 59th Annu. Meeting Assoc. Comput. Linguistics, 11th Int. Joint Conf. Natural Lang. Process., 2021, pp. 3970–3979.
[5] X. Liu, Z. Luo, and H. Huang, “Jointly multiple events extraction via attention-based graph information aggregation,” in Proc. Conf. Empirical Methods Natural Lang. Process., 2018, pp. 1247–1256.
[6] N. Ding, Z. Li, Z. Liu, H. Zheng, and Z. Lin, “Event detection with trigger-aware lattice neural network,” in Proc. Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP), Hong Kong: Association for Computational Linguistics, 2019, pp. 347–356. [Online]. Available: https://aclanthology.org/D19-1033.
[7] X. Wang, X. Han, Z. Liu, M. Sun, and P. Li, “Adversarial training for weakly supervised event detection,” in Proc. NAACL-HLT, 2019, pp. 714–923.
[8] B. Yang and T. M. Mitchell, “Joint extraction of events and entities within a document context,” in Proc. NAACL-HLT, 2016, pp. 289–299.
[9] Y. Lu, H. Lin, X. Han, and L. Sun, “Distilling discrimination and generalization knowledge for event detection via delta-representation learning,” in Proc. 57th Annu. Meeting Assoc. Comput. Linguistics, 2019, pp. 4366–4376.
[10] X. Wang, Z. Wang, X. Han, W. Jiang, R. Han, Z. Liu, J. Li, P. Li, Y. Lin, and J. Zhou, “MAVEN: A massive general domain event detection dataset,” in Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP), 2020, pp. 1–20.
[11] M. Tong, B. Xu, S. Wang, Y. Cao, L. Hou, J. Li, and J. Xie, “Improving event detection via open-domain trigger knowledge,” in Proc. 58th Annu. Meeting Assoc. Comput. Linguistics, 2020, pp. 5887–5897.
[12] A. Kumar, O. Irsy, P. Ondruska, M. Iyyer, J. Bradbury, I. Gulrajani, V. Zhong, R. Paulus, and R. Socher, “Ask me anything: Dynamic memory networks for natural language processing,” in Proc. ICML, 2016, pp. 1378–1387.
[13] M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, and L. Kaiser, “Universal transformers,” in Proc. ICLR, 2019, pp. 1–23.
[14] A. Graves, “Adaptive computation time for recurrent neural networks,” 2016, arXiv:1603.08983.
[15] J. R. Hershey and P. A. Olsen, “Approximating the Kullback Leibler divergence between Gaussian mixture models,” in Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP), Apr. 2007, p. 317.
[16] Y. Zhao, X. Jin, Y. Wang, and X. Cheng, “Document embedding enhanced event detection with hierarchical and supervised attention,” in Proc. 56th Annu. Meeting Assoc. Comput. Linguistics, 2018, pp. 414–419.
[17] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in Proc. NAACL-HLT, 2019, pp. 4171–4186.
[18] V. D. Lai, T. N. Nguyen, and T. H. Nguyen, “Event detection: Gate diversity and syntactic importance scores for graph convolution neural networks,” in Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP), B. Webber, T. Cohn, Y. He, and Y. Liu, Eds. Stroudsburg, PA, USA: Association for Computational Linguistics, Nov. 2020, pp. 5405–5411. [Online]. Available: https://aclanthology.org/2020.emnlp-main.435.
[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 5998–6008.
[20] H. Yan, X. Jin, X. Meng, J. Guo, and X. Cheng, “Event detection with multi-order graph convolution and aggregated attention,” in Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP), 2019, pp. 5766–5770.
[21] S. Cui, B. Yu, T. Liu, Z. Zhang, X. Wang, and J. Shi, “Edge-enhanced graph convolution networks for event detection with syntactic relation,” in Findings of the Association for Computational Linguistics: EMNLP (ACL), T. Cohn, Y. He, and Y. Liu, Eds. Stroudsburg, PA, USA: Association for Computational Linguistics, 2020, pp. 16–20.
[22] D. Lou, Z. Liao, S. Deng, N. Zhang, and H. Chen, “MLBiNet: A cross-sentence collective event detection network,” in Proc. 59th Annu. Meeting Assoc. for Comput. Linguistics 11th Int. Joint Conf. Natural Lang. Process. (EMNLP), B. Webber, T. Cohn, Y. He, and Y. Liu, Eds. Stroudsburg, PA, USA: Association for Computational Linguistics, 2020, pp. 9119–9130.
[23] Y. Meng, C. Xiong, P. Bajaj, S. Tiwary, P. Bennett, J. Han, and X. Song, “COCO-LM: Correcting and contrasting text sequences for language model pretraining,” 2021, arXiv:2102.08473.
[24] B. Yang and T. M. Mitchell, “Joint extraction of events and entities within a document context,” in Proc. NAACL-HLT, 2021, pp. 1–11.
[25] B. Li, H. Zhou, J. He, M. Wang, Y. Yang, and L. Li, “On the sentence embeddings from pre-trained language models,” in Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP), B. Webber, T. Cohn, Y. He, and Y. Liu, Eds. Stroudsburg, PA, USA: Association for Computational Linguistics, 2020, pp. 9119–9130.
[26] Y. Meng, C. Xiong, P. Bajaj, S. Tiwary, P. Bennett, J. Han, and X. Song, “COCO-LM: Correcting and contrasting text sequences for language model pretraining,” 2021, arXiv:2102.08473.
[27] C. Lennon and H. Burdick, “The lexile framework as an approach for reading measurement and success,” Tech. Rep., 2004.
[28] X. Li, X. Wang, X. Han, J. Li, J. Tang, and J. Yang, “Generalized focal loss v2: Learning reliable localization quality estimation for dense object detection,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern
Recognt. (CVPR), Jun. 2021, pp. 11632–11641. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2021/html/Li_Generalized_Focal_Loss_V2_Learning_Reliable_Localization_Quality_Estimation_for_CVPR_2021_paper.html

[27] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, “Focal loss for dense object detection,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Venice, Italy, Oct. 2017, pp. 2999–3007, doi: 10.1109/ICCV.2017.324.

[28] X. Li, Z. Liu, P. Luo, C. C. Loy, and X. Tang, “Not all pixels are equal: Difficulty-aware semantic segmentation via deep layer cascade,” in Proc. IEEE Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 3193–3202.

[29] G. Jawahar, B. Sagot, and D. Seddah, “What does BERT learn about the structure of language?” in Proc. 57th Annu. Meeting Assoc. Comput. Linguistics, A. Korsten, D. R. Traum, and L. Márquez, Eds. Florence, Italy: Association for Computational Linguistics, 2019, pp. 3651–3657.

[30] Z. Jiang, F. F. Xu, J. Araki, and G. Neubig, “How can we know what language models know?” Trans. Assoc. Comput. Linguistics, vol. 8, pp. 423–438, Dec. 2020.

[31] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, “ALBERT: A lite BERT for self-supervised learning of language represenations,” in Proc. 8th Int. Conf. Learn. Represent. (ICLR), Addis Ababa, Ethiopia, 2020, pp. 1–17.

[32] C. Li, G. Wang, B. Wang, X. Liang, Z. Li, and X. Chang, “Dynamic slimmable network,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2021, pp. 8607–8617. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2021/html/Li_Dynamic_Slimmable_Network_CVPR_2021_paper.html

[33] T. Bolukbasi, J. Wang, O. Dekel, and V. Saligrama, “Adaptive neural networks for efficient inference,” 2017, arXiv:1702.07811.

[34] W. Hua, Y. Zhou, C. D. Sa, Z. Zhang, and G. E. Suh, “Channel gating neural networks,” in Proc. Adv. Neural Infor. Process. Syst., H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. B. Fox, and R. Garnett, Eds. Vancouver, BC, Canada, 2019, pp. 1884–1894. [Online]. Available: https://proceedings.neurips.cc/paper/2019/hash/68b1fe71f6e94a302/409f31f23fe3b313-Abstract.html

[35] Z. Chen, Y. Li, S. Bengio, and Y. Bengio, “You look twice: GaterNet for dynamic filter selection in CNNs,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Long Beach, CA, USA, 2019, pp. 9172–9180. [Online]. Available: http://openaccess.thecvf.com/content_CVPR_2019/html/Chen_You_Look_Twice_GaterNet_for_DYNAMIC_FILTER_SELECTION_IN_CNNs_CVPR_2019_paper.html

[36] H.-I. Hu, D. Dey, M. Hebert, and J. A. Bagnell, “Learning anytime predictions in neural networks via adaptive loss balancing,” in Proc. 33rd AAAI Conf. Artif. Intell., AAAI 31st Innov. Appl. Artif. Intell. Conf., IAAI 99 AAAI Symp. Educ. Adv. Artif. Intell., (EAAI), Honolulu, HI, USA, 2019, pp. 3812–3821. [Online]. Available: https://doi.org/10.1609/aaai.v33i01.33018312 HI

[37] W. Liu, P. Zhou, Z. Wang, Z. Zhao, H. Deng, and Q. Ju, “FastBERT: A self-distilling BERT with adaptive inference time,” in Proc. 58th Annu. Meeting Assoc. for Comput. Linguistics. Stroudsburg, PA, USA: Association for Computational Linguistics, Jul. 2020, pp. 6035–6044. [Online]. Available: https://www.aclweb.org/anthology/2020.acl-main.537

[38] A. Heuillet, H. Tabia, H. Arioui, and K. Youcef-Toumi, “D-DARTS: Distributed differentiable architecture search,” 2021, arXiv:2108.09306.

[39] M. Elbayad, J. Gu, E. Grave, and M. Auli, “Depth-adaptive transformers,” in Proc. 38th Int. Conf. Learn. Represent. (ICLR), Addis Ababa, Ethiopia, 2020, pp. 1–15. [Online]. Available: https://openreview.net/forum?id=Slfg7KIYKPH

[40] Y. Liu, F. Meng, J. Zhou, Y. Chen, and J. Xu, “Faster depth-adaptive transformers,” in Proc. 34th AAAI Conf. Artif. Intell., AAAI 33rd Conf. Innov. Appl. Artif. Intell., IAAI 11th Symp. Educ. Adv. Artif. Intell., (EAAI), Virtual Event, 2021, pp. 13424–13432. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/17584

[41] H. Hua, Y. Chen, Y. Liu, Z. Liu, and C. G. Brinton, “On extending NLP techniques from the categorical to the latent space: KL divergence, Zipf’s law, and similarity search,” 2020, arXiv:2012.01941.

[42] M. Labeau and S. B. Cohen, “Experimenting with power divergences for language modeling,” in Proc. Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP), K. Inui, J. Jiang, V. Ng, and X. Wan, Eds. Hong Kong: Association for Computational Linguistics, 2019, pp. 4102–4112, doi: 10.18653/v1/D19-1421.

[43] F. Nielsen, “On a generalization of the Jensen–Shannon divergence and the Jensen–Shannon centroid,” Entropy, vol. 22, no. 2, p. 221, Feb. 2020, doi: 10.3390/e22020221.

[44] Y. Chen and D. Meurers, “Characterizing text difficulty with word frequencies,” in Proc. 11th Workshop Innov. Use NLP Building Educ. Appl. San Diego, CA, USA: Association for Computational Linguistics, Jun. 2016, pp. 84–94. [Online]. Available: https://aclanthology.org/W16-0509

[45] F. Nadeem and M. Ostendorf, “Estimating linguistic complexity for science texts,” in Proc. 13th Workshop Innov. Use NLP Building Educ. Appl., J. R. Tetreault, J. Burstein, E. Kochmar, C. Leacock, and H. Yamakoudakis, Eds. New Orleans, LA, USA: Association for Computational Linguistics, 2018, pp. 45–55. doi: 10.18653/v1/w18-0505

[46] Y.-T. Huang, M. C. Chen, and Y. S. Sun, “Characterizing the influence of features on reading difficulty estimation for non-native readers,” 2018, arXiv:1808.09718.

[47] J. D. Robinson, C. Chuang, S. Sra, and S. Jegelka, “Contrastive learning with hard negative samples,” in Proc. 9th Int. Conf. Learn. Represent. (ICLR), Virtual Event, Austria, May 2021, pp. 1–28. [Online]. Available: https://openreview.net/forum?id=CR1XOOQUTH-

[48] T. H. Nguyen, K. Cho, and R. Grishman, “Joint event extraction via recurrent neural networks,” in Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol., 2016, pp. 300–309.

JUN XU received the bachelor’s degree in the field of computer science and technology and the master’s degree in artificial intelligence from the National University of Defense Technology, China, in 2016 and 2019, respectively. He was awarded a National Scholarship from China, in 2015 and 2018. He was awarded an outstanding dissertation for master’s degree. He was a Meritorious Winner in the 2015 American Mathematical Contest in Modeling. He has been serving as the Senior Algorithm Engineer at Ant Group, since 2019. He has published A Network-embedding Based Method for Author Disambiguation in the International Conference on Information and Knowledge Management (CIKM 2018). His research interests include information extraction, knowledge graph, and machine learning.

MENGSHU SUN received the bachelor’s degree in the field of software engineering and the master’s degree in software engineering from the Beijing Institute of Technology, China, in 2012 and 2015, respectively. She has been serving as the Profficient Engineer at Ant Group, since 2015. She was the Cognitive Computing and Event Logical Graph Team Leader, leading the work on Named Entity Recognition, Relation Extraction, Event Extraction, Entity Linking, and Knowledge Graph Representation. From 2019 to 2021, she received the Best Appy Award on Event Logical Graph, in 2020. She was nominated for Super Ma Award. Her research interests include information extraction, knowledge graph, and knowledge reasoning.