Argument Extraction from Communication Failures via Event Focusing and Terminology Enhancement with a Hybrid Large Language Model Framework

Le Zhang[1,3], Yangke Xu[1], Leihan Zhang[2+]

[1]*School of Computing, Beijing University of Information Technology,*

No. 12, Xiaoying East Road, Qinghe, Haidian District, Beijing

[2]*School of Economics and Management, Beijing University of Posts and Telecommunications,*

No. 10, Xitucheng Road, Beijing

[3]*The State Key Laboratory of Tibetan Intelligence,*

No. 38 Wusi West Road, Chengxi District, Xining City, Qinghai Province

Keywords: Communication Domain; Domain-specific Terminology; Large Language Model; Argument Extraction; Prompt Template

Abstract

Event argument extraction for communication faults involves automatically identifying key elements of fault events from a large number of communication fault documents. This process is crucial for network operation, maintenance, and intelligent fault diagnosis. To tackle the challenges of event co-occurrence and recognizing domain-specific terms in communication fault event argument extraction, this paper proposes an event-focused and terminology-enhanced method. First, a large language model (LLM) is fine-tuned to extract specific event and domain-specific terminology information. Next, an Event Boundary-aware Encoding Module is designed; by embedding the extracted event information into a prompt template and using non-autoregressive encoding, the module helps the model focus on specific event content and improves its ability to distinguish event boundaries. Finally, to better recognize domain-specific terms, a Domain-specific Term Recognition Module facilitates hierarchical semantic interaction between the terms extracted by the LLM and the text, enhancing the model's understanding of these terms. Experiments conducted on the Huawei communication fault dataset and a telecommunication fault dataset show that the proposed method achieves significant improvements over other event argument extraction techniques.

[+]Corresponding author: Leihan Zhang (Email: zhangleihan@gmail.com; ORCID:).

1. Introduction

With the rapid development of information technology, the scale of communication networks has grown significantly, leading to a corresponding rise in the frequency and impact of faults. Timely and accurate detection and management of fault events are essential for maintaining the stable operation of communication networks. However, when dealing with large volumes of textual data, such as network logs and alarm reports, traditional manual fault analysis methods become inefficient and cannot meet the demands of modern communication systems. Event Extraction (EE), an important area of research in Natural Language Processing (NLP), focuses on identifying and extracting events and their related participants from text [1], offering new possibilities for automated communication fault analysis. The Event Extraction (EE) task mainly involves two sub-tasks: Event Detection (ED) and Event Argument Extraction (EAE) [2]. ED aims to locate trigger words within a sentence and classify their corresponding event types. EAE, meanwhile, concentrates on finding the positions of arguments in the sentence and classifying their roles. Figure 1 shows an example sentence with one trigger word and three arguments. The ED task is to identify the trigger word "is adjusted" and categorize it as the "Machine Adjustment" event type. The task of EAE is to identify the spans of arguments in a sentence and classify them into specific argument roles, such as "FinalState" and "Object".

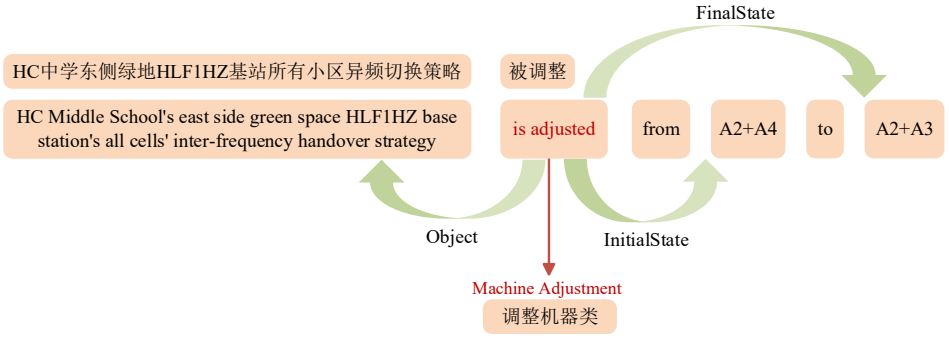

Figure 1. Example of event extraction for the sentence: HC Middle School's east side green space HLF1HZ base station's all cells' inter-frequency handover strategy is adjusted from A2+A4 to A2+A3.

In recent years, research on EAE has made significant progress, mainly due to advances in deep learning. However, most of these studies tend to ignore the issue of event co-occurrence, where multiple events appear within the text. This oversight often results in models having a weaker ability to identify event boundaries. At the same time, regarding the challenge of accurately recognizing domain-specific terminology in the communication field, existing methods generally lack focused processing strategies. As a result, argument content related to these domain-specific terms is often hard to identify accurately, which negatively impacts the performance of Event Argument Extraction.

To address these limitations, this paper proposes an EFTE (Event Boundary-aware Encoding and Semantically-enhanced Domain-specific Term Recognition) model, a novel method capable of enhancing inter-event boundary perception in multi-event contexts while simultaneously improving the recognition accuracy of terminological arguments. The contributions of this paper are summarized as follows:

(1) To address the issue of event co-occurrence, EFTE differs from current methods that extract all events in the text at once. Instead, it concentrates on extracting arguments for only one event type at a time. Afterwards, it improves its understanding of event boundaries during extraction by designing an Event Boundary-aware Encoding Module and a Term Recognition Module.

(2) We designed an Event Boundary-aware Encoding Module. By directing the model's attention to specific events within the text during the encoding phase, this module enhances its ability to discriminate between event boundaries, thereby addressing the issue of event co-occurrence.

(3) We propose a Semantically-enhanced Domain-specific Term Recognition Module. This module enhances its recognition of domain-specific terms by interactively fusing the terms extracted by an LLM at multiple levels.

2. Related Work

Since Liu et al. [3] formally introduced prompt learning to the broader research community, EAE methods based on prompt templates emerged. Owing to their superior performance and generalization capabilities, these methods have become one of the mainstream techniques in this field [4][5]. According to the different ways arguments are extracted, current prompt-template-based EAE methods are primarily categorized into prompt-learning-based classification methods and prompt-learning-based generative methods. Within prompt-learning-based classification, researchers have pursued optimizations across multiple dimensions. For instance, in terms of template design and interaction, Ma et al. [6] captured interactions between argument roles via multi-role prompts and employed joint optimization for global assignment. Meanwhile, Xue et al. [8] explored automatic template generation to alleviate the burden of manual construction. As for model efficiency, Huang et al. [7] boosted inference speed by optimizing the pre-trained model's hierarchical structure. To compensate for the limitations of the aforementioned methods in semantic understanding, Yang et al. [9] integrated entity type information of argument roles into templates. By employing deep encoding, they deepened the model's understanding of argument role semantics, thereby improving the model's ultimate performance. These works have significantly advanced the development of classification-based EAE methods, boosting their performance in various scenarios. Concurrently, generative methods have also made progress on specific challenges, such as achieving cross-sentence extraction [10], cross-lingual transfer [11], and performance improvements in low-resource settings [12].

Although the methods above excel in template design, semantic understanding, and scenario-specific adaptation, they typically lack a specialized mechanism for handling event co-occurrence. The common practice is implicitly linking an event to its trigger word for localization [13], an approach that introduces considerable uncertainty and struggles to define event boundaries precisely. Recognizing this limitation, subsequent research has focused on leveraging inter-event correlations to delineate these boundaries better. For instance, Hu et al. [14] and Liu et al. [15] utilized structure-aware attention mechanisms to connect multiple events, enhancing the model's ability to discern semantic scopes. To this end, Peng et al. [15] also proposed a prefix module that integrates features from co-occurring events into a pre-trained model's attention calculations to process inter-event relationships better. By focusing on these relationships, such methods have progressed in addressing the challenges of event co-occurrence.

However, in communication fault data, different events often show relative independence, their relationships are relatively simple, and there is a high use of domain-specific terminology, which causes semantic ambiguity in the text, as shown in Figure 2. Two events may share the same trigger word but lack complex relationships between them, and their arguments may not overlap. As a result, accurately distinguishing such events using only trigger words or relationships becomes difficult. At the same time, current methods often lack specialized strategies and solutions for processing data with domain-specific terminology, limiting the effectiveness of event extraction in the communication domain. Therefore, better handling of domain-specific terminology and improving the model's ability to identify event boundaries remain key challenges for event extraction tasks in the communication domain.

In recent years, with the rapid development of artificial intelligence technology, LLM have increasingly become a key focus of research and application [16]. Due to their powerful computational abilities and training on large datasets, LLM have made significant progress across various tasks [17][18], opening new opportunities to solve challenges in EAE within the communication domain. In this paper, we propose an event boundary-aware event argument extraction method by integrating LLM, which significantly enhances the model's ability to perceive event boundaries and recognize domain-specific terms.

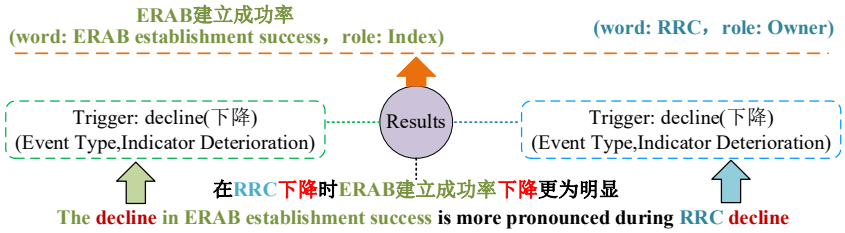

Figure 2. The process of extracting event arguments from a text that contains multiple events. In this example, the two events have the same event type and trigger word. Relying only on trigger words and event types, traditional argument

extraction methods find it difficult to identify the boundaries of each event.

3. Method

The EFTE method includes a Large Language Model Fine-tuning Module, an Event Boundary-aware Encoding Module, a Semantically-enhanced Domain-specific Term Recognition Module, and an Attention-Guided Argument Extraction Module. The model architecture is shown in Figure 3. Each module will be explained in detail in the following subsections.

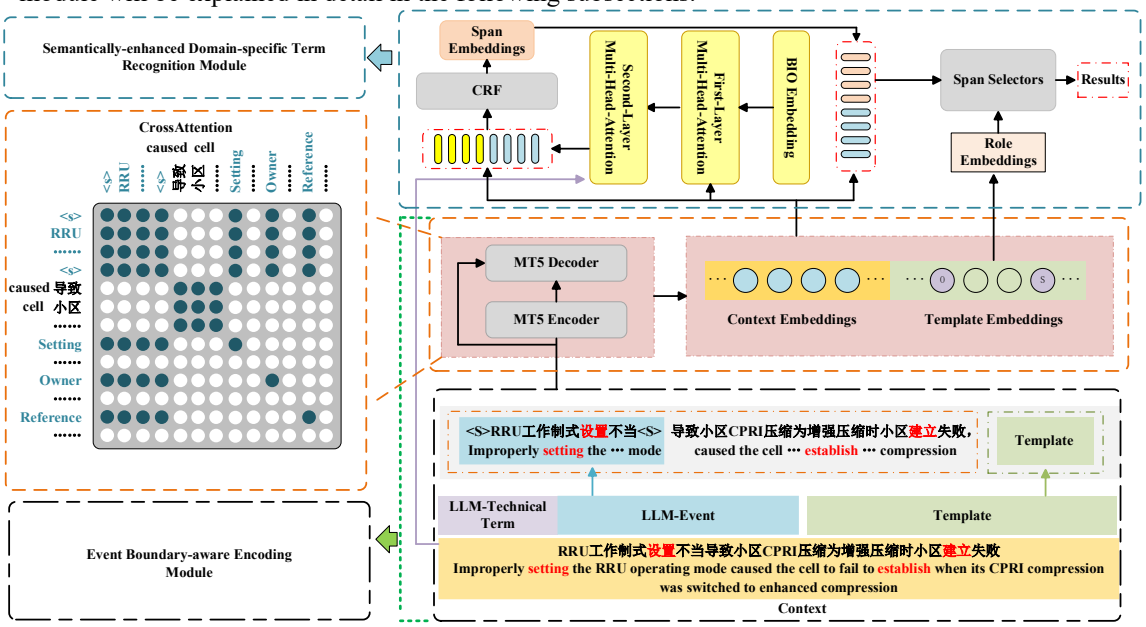

Figure 3. The architecture of EFTE. The model is shown processing a target event (highlighted in blue) to extract its corresponding arguments.

3.1 Large Language Model Fine-tuning Module

Applying an LLM pre-trained on general corpora directly to specific domains often produces unsatisfactory results [19]. Fine-tuning can improve the LLM's adaptability to particular domains and enhance task performance. For the fine-tuning process, the Low-Rank Adaptation (LoRA) [20] method is used to adjust the LLM. Regarding the prompting strategy, the Chain-of-Thought (CoT) [21] approach is adopted, allowing the LLM to extract events and their corresponding domain-specific terminology simultaneously. The Qwen2.5-instruct-14B [22] model was chosen as the LLM.

The fine-tuning tasks are primarily divided into event span extraction and domain-specific terminology extraction. To fine-tune for these tasks, we need to annotate training data for both parts. For the annotation of the event span extraction task, we identify event boundaries by finding the highest and lowest indices of all argument spans within each event. The text within these boundaries is then used as labels for the event span extraction task. For the annotation of the domain-specific terminology extraction task, we conducted statistical analysis and summarization of the argument content and role types in the dataset. Ultimately, different argument roles from the original dataset were categorized into three major types of domain-specific terms. These categories become the labels for the domain-specific terminology extraction task. As shown in Figure 4, "Owner", "Subject", etc., represent the original argument roles in the dataset, while "Fault Name", "Fault Status", etc., are the grouped domain-specific terminology types. Because the label annotation for both tasks builds upon existing annotations in the original dataset, we were able to streamline the annotation process without requiring extensive, separate consistency checks to ensure quality.

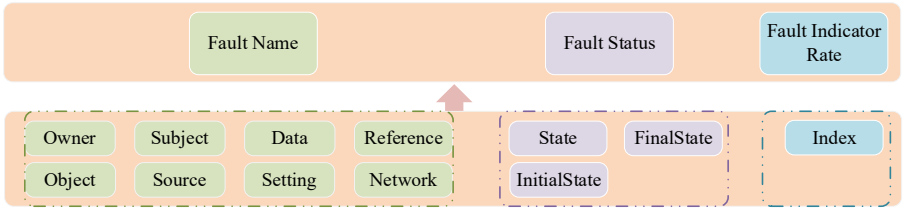

Figure 4. The proposed labeling scheme for fine-tuning Large Language Models on the domain-specific term extraction task. To support the fine-tuning process, the original argument roles are recategorized into three distinct classes.

After preparing the training labels needed for the fine-tuning tasks, a prompt template based on the CoT method is designed to fine-tune the LLM. This prompt template, as shown in Figure 5, can extract event spans while also identifying the domain-specific terminology within them. The prompt template performs event span extraction by combining the event type and trigger word. After successfully extracting the event span linked to the trigger word, it further adds prompt statements to continue extracting domain-specific terminology from the event span.

You are an expert in the field of event extraction. Below are some examples of event extraction. You need to follow the process outlined to extract events from the given text.
First, you need to determine if one or more events exist in the text (e.g., "Unreasonable configuration of 3G-side reselection threshold leads to an increase in L2U coverage redirection"). Then, for the event types and trigger word pairs I provide, such as: [Event Type:Configuration-related Fault-Trigger Word:configuration, Event Type:Software/ Hardware Anomaly-Trigger Word:redirection], you are to extract the corresponding events.
For example, based on the text above and the two provided pairs, two events can be obtained: [{'Event Content':'Unreasonable configuration of 3G-side reselection threshold','Event Type':'Configuration-related Fault'},{'Event Content':'increase in L2U coverage redirection','Event Type':'Software/Hardware Anomaly'}] (Note: The extracted event must be a continuous span of text from the original passage and cannot be arbitrarily generated.)
Next, you also need to extract the fault entities from the extracted event texts. There are three types of fault entities: Fault Name, Fault Status, and Fault Index Rate. For instance, the fault entities that can be extracted from the two events above are: [[{'word':'3G-side reselection threshold','type':'Fault Name'},{'word':'unreasonable','type':'Fault Status'}],[{'Fault Name':'L2U coverage','Fault Status':'increase'}]]
Based on the event extraction examples above, extract the events from the following text: "Pushing a low-version license item directly to a high-version one caused the license application to fail." The event type and trigger word pairs involved in the text are: [Event Type:Software/Hardware Anomaly-Trigger Word:application, Event Type:Machine Adjustment-Trigger Word:pushing]

Figure 5. The prompt template is created for fine-tuning large language models on a two-step extraction task. This template uses a CoT approach, guiding the model first to identify event spans and then to extract the related domain-specific terms.

3.2 Event Boundary-aware Encoding Module

In contrast to approaches that tackle event co-occurrence through inter-event correlations, the Event Boundary-aware Encoding module fuses non-autoregressive encoding with event embeddings. This fusion preserves the integrity and independence of each event, leading to a strengthened perception of their boundaries. Specifically, this method first embeds event information generated by a fine-tuned LLM into a prompt template, which is then processed by a non-autoregressive encoder. During the encoding stage, the model uses the "<s>" marker and a self-attention mechanism to accurately locate the target event for the argument being extracted. Afterward, in the cross-attention calculation during the decoding stage, the model employs the "<s>" token as a central anchor. This anchor dynamically guides the attention of features toward the indicated target event, such as the attention of argument roles like 'Owner' (as shown in Figure 3) being focused on the "<s> Improperly setting the … mode <s>" event. This design ultimately achieves efficient aggregation of specific event-related information, thereby significantly enhancing the model's understanding of event boundaries.

When constructing the prompt template focused on a specific event, first, the event span generated by the LLM is precisely located within the text and explicitly marked with a specific tag "<s>" (the annotation effect is illustrated in the "LLM-guided Event Focusing Prompt Template Module" section of Figure 3). Building on this, and drawing inspiration from the method of Yang et al. [9], who embedded argument entity types into prompt templates, corresponding prompt templates were designed for the seven major categories of fault events in the dataset (as shown in Table 1), with domain-specific terminology types

embedded within them. Finally, the event text (with explicit event span annotation) is concatenated with the prompt template text of the respective event type to create the final prompt template content for input to the encoder. The prompt template is formulated as follows:

$$[< s > E_1 < s > E_2, \ldots P; J; W] \quad (1)$$

Here, E_i represents the i-th event span contained in the text, P denotes the remaining string information from the source text, J is the event type prompt prefix constructed for different training samples, which is primarily composed of the event type and trigger word corresponding to the given data. W is the template for the specific event type detailed in Table 1.

Table 1. Event Category Template Table

Event Category	Template
Indicator Deterioration	Owner (Fault Name) has an Index (Fault Indicator Rate) that produces a State (Fault Status).
Software/Hardware Anomaly	Subject (Fault Name) or Owner (Fault Name) causes State (Fault Status) on Object (Fault Name).
Data Collection	Source (Fault Name) collects Data (Fault Name).
Verification	Owner (Fault Name) checks Object (Fault Name).
Configuration Fault	Setting (Fault Name) or Owner (Fault Name) configuration is incorrect, reference object is Reference (Fault Name).
Machine Adjustment	Network (Fault Name) adjusts the machine Object (Fault Name) state from InitialState (Fault Status) to FinalState (Fault Status).
Machine Operation	Owner (Fault Name) operates on Object (Fault Name).

Since domain-specific terminology in communication datasets often appears in English, this paper chooses the multilingual version of the T5 model (Multilingual T5, MT5) [23] as the encoder, allowing it to understand the semantics of both English and Chinese words fully. EFTE improves the interaction between event text and the template by non-autoregressively encoding the constructed prompt template text with MT5, as shown below:

$$[C, T] = CompositeMT5Encoder(X),\ C \in R^{N \times d_t},\ T \in R^{M \times d_t} \quad (2)$$

This describes a composite encoding architecture. Let X be the input that is simultaneously fed into the MT5 encoder and a non-autoregressive decoder, ultimately yielding the encoded textual features C and T. Here, C represents the encoded features of the original text, and T represents the encoded features of the prompt template. N and M are the lengths of the original text and the prompt template, respectively, and d_t is the encoding dimension of the vectors.

3.3 Semantically-enhanced Domain-specific Term Recognition Module

Addressing the prevalence of domain-specific terminology in communication fault datasets, EFTE incorporates an LLM to design a semantically-enhanced domain-specific terminology module. The prediction of terminology is essentially a sequence labeling task. Yang et al. [9], in their effort to achieve a deeper understanding of argument semantics, integrated the entity types of arguments into the EAE process. Drawing inspiration from this approach, an embedding matrix for domain-specific terminology types is constructed based on the event types found in communication domain data, as shown below:

$$B = BIOEmbedding(L \cdot 3, d_t),\ B \in R^{(L \times 3) \times d_t} \quad (3)$$

Here, L represents the number of event types in the dataset, and 3 signifies the token taggin g types, which can be B (Beginning), I (Inside), or O (Outside) tags.

After creating the BIO embedding layer, the embedding matrix B and the textual features C and T (obtained from the encoder, as previously described) are interactively fused using a multi-head attention mechanism. This process aligns the terminology tagging features in the embedding matrix B with the terminology-related features in C and T. The calculation is performed as follows:

$$Q = [C, T]W_q,\ W_q \in R^{d_t \times d_t} \quad (4)$$

$$K = B'W_k,\ W_k \in R^{d_t \times d_t} \quad (5)$$

$$V = B'W_v,\ W_v \in R^{d_t \times d_t} \quad (6)$$

$$O_1 = CMA(Q,K,V),\ O_1 \in R^{N\times d_t} \tag{7}$$

Here, B' is the feature vector derived from B after reshaping or expanding its dimensions to match the batch size of other features, preparing it for subsequent fusion. W_q, W_k, and W_v are linear transformation weight matrices, and the function H represents a multi-head cross-attention network layer.

After the first layer of the attention mechanism, the BIO embedding layer has achieved an initial perception and alignment of terminology within the C (original text features) and T (template features) vectors. To further enhance the perception of domain-specific terminology, a dual-layer attention fusion mechanism is designed. The second layer of the attention mechanism strengthens the semantic representation of domain-specific terminology in the fused feature vectors by incorporating the domain-specific terminology generated by the LLM, as shown below:

$$Q = [C,T]W_q,\ W_q \in R^{d_t\times d_t} \tag{8}$$

$$K = ZW_k,\ W_k \in R^{d_t\times d_t} \tag{9}$$

$$V = ZW_v,\ W_v \in R^{d_t\times d_t} \tag{10}$$

$$O_2 = CMA(Q,K,V),\ O_2 \in R^{N\times d_t} \tag{11}$$

Here, Z represents the domain-specific terminology generated by the LLM, after its features have undergone shape expansion. Finally, the features resulting from the dual-layer attention mechanism fusion are concatenated with C (original text features) and T (template features) to obtain the final semantic vector used for domain-specific terminology prediction, as shown in the following equation:

$$O = Concat(O_1, O_2, O_2 \cdot [C,T], O_1 \cdot [C,T]),\ O \in R^{N\times 5\times d_t} \tag{12}$$

$$O = Linear\_one(O),\ O \in R^{N\times d_t} \tag{13}$$

$$O = Linear\_second(O),\ O \in R^{N\times 3\times L} \tag{14}$$

Finally, Conditional Random Fields (CRF) [24] are used for sequence labeling, and the log-likelihood function is taken as the loss function for the technical term prediction module. The loss is denoted as $Loss_{ner}$.

3.4 Attention-Guided Argument Extraction Module

For the argument extraction module, we propose an attention-focused argument extraction scheme. By deeply coupling the argument extraction module with the Event Boundary-aware Encoding Module, we ensure that the semantic information of argument roles is concentrated on the specific target event during the encoding stage, thereby achieving focused attention throughout the argument extraction process. As shown in Figure 3, when extracting arguments for the event "Improperly setting the RRU operating mode caused the cell to fail to establish when its CPRI compression was switched to enhanced compression", the argument roles (e.g., Owner, Setting) focus their attention on "Improperly setting the RRU operating mode" during the encoding stage. Furthermore, to further enhance the accuracy of argument selection, we draw inspiration from the work of Yang et al. [9]. They significantly optimized decoding results by making joint decisions from a dual perspective of "argument role" and "entity type". Inspired by this approach, we also incorporate a dual-view optimization mechanism, leveraging domain knowledge (specifically, specialized terminology) to aid the decision-making process and achieve more precise decoding results. The overall algorithm is shown in Algorithm 1.

The encoded event type template feature vector T contains all the argument roles for a specific event type, as detailed in Table 1. The features corresponding to different roles within T are extracted, and span selectors, S_{rolei} and E_{rolei}, are constructed for each role to identify the start and end positions of that argument role in the text, as shown below:

$$S_{rolei} = EmbeddingT(role_i) * W_{start},\ W_{start} \in R^{1\times d_t} \tag{15}$$

$$E_{rolei} = EmbeddingT(role_i) * W_{end},\ W_{end} \in R^{1\times d_t} \tag{16}$$

Here, $EmbeddingT(role_i)$ represents the encoded feature corresponding to the i-th argument role in T. W_{start} and W_{end} are learnable weight matrices shared across all argument roles. $*$ represents element-wise multiplication.

The encoded features C of the event text to be extracted and the final features O from the domain-specific terminology prediction module undergo a linear dimensionality transformation to obtain the final hidden features F, as shown in Equations (17-18):

$$F = Concat(O,C),\ F \in R^{N\times(d_t\times d_t)} \tag{17}$$

$$F = FW_f,\ W_f \in R^{(d_t+d_t)\times d_t} \tag{18}$$

Based on the aforementioned features, the probability distributions for the start and end positions of each argument role are calculated, as shown below:

$$SLogit_{rolei} = Sigmoid(S_{rolei} \cdot F) \tag{19}$$

$$ELogit_{rolei} = Sigmoid(E_{rolei} \cdot F) \tag{20}$$

After obtaining the probability distributions for different argument roles, to facilitate joint dec ision-making for argument selection using the dual-perspective semantic understanding mechanism (considering both argument roles and domain-specific terminology), it is first necessary to acqui re the probability distributions for the start ($SLogit_{termi}$) and end ($ELogit_{termi}$) positions of the d omain-specific terminology. Analogous to the method for obtaining argument role probability dist ributions, these probability distributions for domain-specific terminology are then multiplied by th e argument role probability distributions. This product serves as the final probability distribution for EAE, as shown below:

$$FSLogit_{rolei} = SLogit_{rolei} \cdot SLogit_{termi} \tag{21}$$

$$FELogit_{rolei} = ELogit_{rolei} \cdot ELogit_{termi} \tag{22}$$

The final loss for EAE is calculated using the cross-entropy loss function between the predicted probability distributions $FSLogit_{rolei}$ and $FELogit_{rolei}$ and the ground-truth distributions y_{rolei}^{start} and y_{rolei}^{end}, as shown in Equations (23-24):

$$L_{eae} = \sum_{k=1}^{D}\sum_{i=1}^{R} L_{bce}(FSLogit_{rolei}(k), y(k)_i^{start}) + L_{bce}(FELogit_{rolei}(k), y(k)_i^{end}) \tag{23}$$

$$L_{bce}(x, y) = -(ylogx + (1-y)\log(1-x)) \tag{24}$$

Where D is the dataset, and R is the set of argument roles in the current data.

The loss for EAE and the loss for domain-specific terminology prediction are combined to form a joint loss. This joint loss is then backpropagated at each training step to update the model parameters, as shown below:

$$L_{loss} = L_{eae} + L_{ner} \tag{25}$$

Algorithm 1 Argument Extraction

Input: T, C, O // Template features, original text features, terminology features

Output: $FSLogit_{role}$, $FELogit_{role}$ // Extract the feature distribution of argument start and end positions

1: **for** each $rolei$ in T **do**

2: $S_{rolei} = EmbeddingT(role_i) * W_{start}$

3: $E_{rolei} = EmbeddingT(role_i) * W_{end}$

4: **end for**

5: **Fusion Operation:**

6: $F = Concat(O, C),\ F \in R^{N\times(d_t\times d_t)}$

7: $F = FW_f,\ W_f \in R^{(d_t+d_t)\times d_t}$

8: **for** each $rolei$ in T **do**

9: $SLogit_{rolei} = Sigmoid(S_{rolei} \cdot F)$

10: $ELogit_{rolei} = Sigmoid(E_{rolei} \cdot F)$

11: **end for**

12: $T \rightarrow O$, Re-Executing 1-11, $\rightarrow SLogit_{term},\ ELogit_{term}$ // Replace T with O and re-execute steps 1-11 to obtain the feature distribution of the extracted term start and end positions.

13: **Fusion Operation:**

14: $FSLogit_{role} = SLogit_{role} \cdot SLogit_{term}$

15: $FELogit_{role} = ELogit_{role} \cdot ELogit_{term}$

16: **end for**

4. Experiments

4.1 Datasets

To verify the practical effectiveness of the EFTE method in the communication fault domain, experiments were performed on the publicly available Huawei communication fault dataset and a telecommunication fault dataset. Table 2 presents the partitioning of these two datasets into training and testing sets.

The datasets are introduced as follows:

(1) Huawei Communication Fault Dataset: This dataset comes from the "CCKS-2021 Huawei Process-oriented Event Extraction for the Communication Domain" task. It is a publicly available event extraction dataset in the communication fault domain. Fault events in this dataset were defined by Huawei experts based on various types of process-oriented knowledge within the communication domain, and they play a crucial role in analyzing critical errors that happen during network operation and maintenance processes.

(2) Telecommunication Fault Dataset: This dataset is based on real fault data generated during internal network operation and maintenance processes within a telecommunication company. Building on this, we created a high-quality dataset designed explicitly for the EAE task.

Table 2. Statistics of the Datasets

Dataset Split	Huawei Communication Fault Dataset	Telecommunication Fault Dataset
Training Set	11000	5500
Validation Set	2000	1000
Test Set	2000	1000
Total	15000	7500

4.2 Implementation Details

To ensure the reproducibility of our model, this paper provides a detailed description of the experimental environment setup and model parameter settings.

We implement EFTE with Pytorch and run the experiments with a Nvidia Tesla V100 GPU. We set the maximum input length of the encoder to 220 tokens. Sentences exceeding this maximum are truncated to the first 220 tokens, while shorter sentences are padded with the <PAD> token. The detailed parameter settings for the EFTE model are provided in Table 3. Additionally, specific experimental environment parameter settings are detailed in Appendix A.

Table 3 Model Parameter Settings

Hyperparameter	Value
Learning Rate	4e-5
Train Batch Size	8.0
Eval Batch Size	16.0
Epochs	50.0
Dropout	0.1
Gradient Accumulation Steps	1.0
Max Grad Norm	5.0
Seed	42.0

4.3 Evaluation Metrics

The model's performance is evaluated using two standard metrics for Event Argument Extraction (EAE). For each of these metrics, precision, recall, and F1 score [25] are used to assess performance:

(1) Argument Identification (AI): An argument is correctly identified when its predicted span matches the ground-truth span.

(2) Argument Classification (AC): An argument is correctly classified when its predicted span matches the ground-truth span and its predicted argument role is also correct.

4.4 Experimental Results and Analysis

In this experimental section, the EFTE method is compared with EAE methods from the past five years on both the telecommunication fault dataset and the Huawei communication fault dataset. Specifically, to demonstrate the superiority of the proposed method, comparisons were made against representative classification-based and generative-based EAE approaches. The performance of the models on the Huawei communication fault dataset and the telecommunication fault dataset is shown in Tables 4 and 5. In these tables, bold values indicate the best result in each respective column of the experimental data.

The baseline methods used for comparison are listed below:

EEQA [26]: This method proposes transforming the event extraction task into a question-answering task. By designing question templates, it converts ED and EAE into processes of asking questions about the input sentence and returning the event extraction results as answers.

BartGen [10]: This is a document-level EAE model based on conditional generation. The method first encodes event templates (with unfilled argument parameters) and the context text. It then fills the argument parameters in the templates through conditional generation, ultimately achieving the parsing and extraction of event arguments.

XGear [11]: This method proposes an approach based on multilingual pre-trained models by designing language-agnostic event templates for argument parameter filling. By transforming the EAE task into a language generation task, it achieves zero-shot cross-lingual extraction of event argument parameters.

PAIE [6]: This method adds argument roles to prompt templates and captures interactions between parameters through multi-role prompts. It employs joint optimization with a bipartite matching loss to achieve optimal argument span assignment, enabling the model to achieve better learning and generalization with limited training data.

DEGREE [12]: This method formulates the EAE task as a conditional generation problem. Given context text and a designed prompt template, the model summarizes events in the text into natural sentences conforming to a predefined pattern. Finally, it extracts argument parameters by parsing these sentences. By utilizing label semantics and weak supervision signals, it improves the model's learning efficiency in low-resource scenarios.

TagPrime [7]: This method provides a unified processing framework for tasks such as event extraction and relation extraction. For the EAE task, it enhances the input text by adding event types and trigger words. It leverages the self-attention mechanism of pre-trained models, enabling the model to better capture relational structures under specific conditions, and accelerates the EAE inference process through parallel encoding.

TabEAE [14]: This method proposes an EAE approach that uses an attention mechanism to strengthen connections between different events. Unlike previous research that only considered single events, this model simultaneously extracts all argument parameters for each event during training. This definition considers all co-occurring events within the same context, enabling the model to better capture the semantic boundaries of events and improve its multi-event processing capabilities.

DEEIA [5]: Similar to TabEAE, the DEEIA method is an event extraction approach that addresses the multi-event problem. It first utilizes an attention mechanism to strengthen both intra-event and inter-event connections, and then employs a specific-event information aggregation module to enhance its capability for extracting specific events.

Scented [9]: This model customizes the integration of argument entity type information into EAE by incorporating argument roles, argument entity types, and their correspondences into prompt templates. This effectively addresses issues present in existing methods, such as weak semantic relevance, compromised semantic integrity, and one-sided semantic understanding.

Among the methods mentioned earlier, the performance of the EEQA model was relatively below expectations. This is mainly due to its design that extracts arguments using a question-answering approach, which requires strong semantic understanding from the model. However, data in the communication fault domain is known for its strong domain-specificity and semantic ambiguity. Compared to datasets from other areas, understanding these data proves much more difficult, which directly limits the effectiveness of Event EAE. In contrast, the proposed EFTE model achieved better results in both Recall and F1-score.

Specifically, on the publicly available Huawei communication fault dataset, compared to BartGen, the best-performing generative method, EFTE achieved improvements of 9.76 and 8.64 percentage points in

Recall, and 11.51 and 10.30 percentage points in F1-score, for the AI and AC metrics, respectively. On the telecommunication fault dataset, EFTE improved Recall by 4.90 and 4.97 percentage points, and F1-score by 7.64 and 7.58 percentage points, for the AI and AC metrics, respectively.

Compared to Scented, the best-performing classification-based method, on the publicly available Huawei communication fault dataset, EFTE improved Recall by 6.64 and 6.29 percentage points, and F1-score by 2.92 and 2.71 percentage points, for AI and AC, respectively. On the telecommunication fault dataset, EFTE improved Recall by 4.88 and 5.56 percentage points, and F1-score by 2.01 and 2.80 percentage points, for AI and AC, respectively. This demonstrates that the EFTE method achieves excellent performance in EAE.

This significant performance improvement is primarily attributed to two key designs of the model: firstly, the event-focusing module effectively enhanced the model's ability to distinguish boundaries when processing multi-event samples; secondly, the domain-specific terminology prediction module, by accurately predicting domain-specific terms within events, improved the model's accuracy in domain-specific terminology recognition. These innovative designs collectively contributed to the significant improvement in the model's performance on the evaluation metrics.

Table 4. Performance Metrics of Different Algorithm Models on the Huawei Dataset

Model	AI-P	AI-R	AI-F1	AC-P	AC-R	AC-F1	Type
BartGen	70.91	73.95	72.40	68.40	71.33	69.83	Generative
XGear	84.58	62.03	71.57	81.47	59.74	68.93	Generative
DEGREE	61.73	68.59	64.98	54.28	60.32	57.15	Generative
EEQA	44.92	53.35	48.77	41.73	49.56	45.31	Classification
PAIE	78.79	78.36	78.58	74.96	74.74	74.85	Classification
TabEAE	65.05	65.92	66.82	63.66	62.19	62.92	Classification
DEEIA	68.22	66.22	67.21	64.79	63.09	63.93	Classification
TagPrime	78.04	78.79	78.41	73.97	74.68	74.32	Classification
Scented	**84.51**	78.46	81.38	**80.78**	75.02	77.80	Classification
EFTE	84.12	**83.71**	**83.91**	80.29	**79.97**	**80.13**	Classification

Table 5. Performance Metrics of Different Algorithm Models on the Telecommunication Dataset

Model	AI-P	AI-R	AI-F1	AC-P	AC-R	AC-F1	Type
BartGen	73.73	77.05	75.36	71.08	74.27	72.64	Generative
XGear	85.63	61.40	71.52	82.23	59.67	69.50	Generative
DEGREE	54.58	61.73	57.94	51.23	57.94	54.38	Generative
EEQA	44.64	54.03	48.89	42.22	51.10	46.24	Classification
PAIE	80.10	77.81	78.93	76.70	75.02	75.85	Classification
TabEAE	66.20	67.32	66.76	64.53	62.80	63.64	Classification
DEEIA	70.32	69.82	70.08	64.92	65.30	65.10	Classification
TagPrime	80.61	78.84	79.71	77.41	75.72	76.56	Classification
Scented	**85.33**	77.07	80.99	**81.57**	73.68	77.42	Classification
EFTE	84.06	**81.95**	**83.00**	81.22	**79.24**	**80.22**	Classification

4.5 Ablation Study

The core architecture of the EFTE method includes two main modules: the Event Boundary-aware Encoding Module and the Semantically-enhanced Domain-specific Terminology Recognition Module. To assess the effectiveness of each module and its contribution to overall performance, they were individually evaluated and analyzed through ablation studies. The outcomes of these ablation experiments on the publicly available Huawei communication fault dataset and the telecommunication fault dataset are shown in Tables 6 and 7.

The ablation settings are described as follows:

w/o-Terminology-LLM-Fusion: This indicates the model's performance after removing the domain-specific terminology module (which includes the LLM-based terminology extraction and fusion) from the full EFTE model.

w/o-Event-Aware-Encoder: This shows the model's performance after removing the Event Boundary-aware Encoding Module from the complete EFTE model. (To demonstrate the effectiveness of the event boundary-aware encoder, the specially designed event-focusing prompt template within the encoding module was replaced with a generic prompt template, thereby systematically removing the model's ability to recognize specific event boundaries.)

Table 6. Ablation Study Results of the EFTE Method on the Huawei Dataset

No.	Model	AI-F1/%	AC-F1/%
1	w/o-Terminology-LLM-fusion	83.25	79.53
2	w/o-Event-Aware-Encode	82.34	78.73
3	**EFTE**	**83.91**	**80.13**

Table 7. Ablation Study Results of the EFTE Method on the Telecommunication Dataset

No.	Model	AI-F1/%	AC-F1/%
1	w/o-Terminology-LLM-fusion	82.52	79.63
2	w/o-Event-Aware-Encode	81.65	78.90
3	**EFTE**	**83.00**	**80.22**

The results from Experiment 1 (w/o-Terminology-LLM-Fusion) reveal that after removing the domain-specific terminology prediction module, the AI-F1 and AC-F1 scores of EFTE decreased by 0.66 and 0.60 percentage points on the Huawei dataset, and by 0.48 and 0.59 percentage points on the telecommunication dataset, respectively. This indicates that the proposed domain-specific terminology module, which incorporates an LLM and a dual-layer attention mechanism, positively improves the performance of EAE.

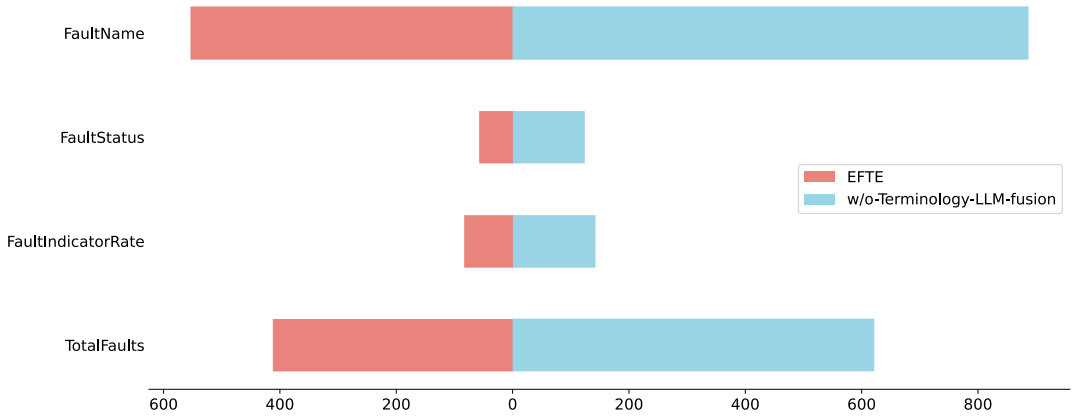

Figure 6. Distribution of Error Types in Technical Term Extraction

In Experiment 2 (w/o-Event-Aware-Encoder), removing the event-focusing prompt template component from the Event Boundary-aware Encoding Module led to a decrease in AI-F1 and AC-F1 scores by 1.57 and 1.40 percentage points on the Huawei dataset, and by 1.35 and 1.32 percentage points on the telecommunication dataset. This notable performance drop confirms the importance of the Event Boundary-aware Encoding Module.

To investigate the effectiveness of the proposed modules further, we conducted an in-depth comparative analysis of each. Specifically, in the Domain-specific Term Recognition Module analysis, the focus was on comparing the distribution and count of technical term extraction errors after integrating the EFTE module, as shown in Figure 6. We observe a significant reduction in the overall number of term extraction errors (TotalFaults) with the application of this module. Among the three term types, "Fault

Name" accounts for a relatively higher number of errors, which can be attributed to the prevalence of this term type within the primary communication terminology. Nevertheless, EFTE significantly reduced extraction errors across all three categories. This demonstrates the module's exceptional effectiveness in mitigating term extraction errors.

In the comparative analysis of the Event Boundary-Aware Encoding Module, we focused on errors caused by arguments crossing event boundaries. Table 8 details the distribution of multi-event samples, and Figure 7 presents the results on the Huawei test set. After integrating this module, we observed significant improvements. The number of cross-boundary errors decreased from 309 to 130, reducing the proportion of total errors from 58.9% to 38.0%. Furthermore, the total count of extraction errors was reduced from 525 to 342. These results validate that our proposed module enhances the model's sensitivity to event boundaries while improving its overall extraction performance.

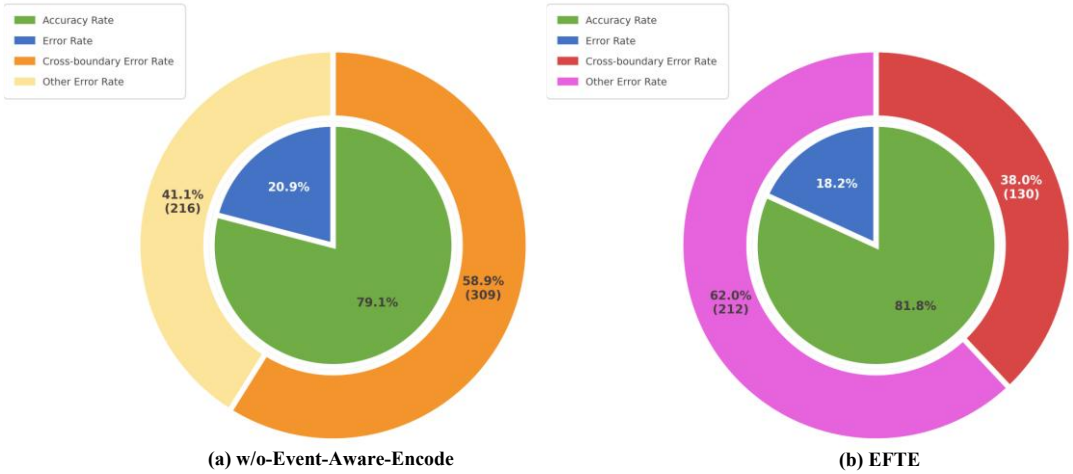

Figure 7. Statistical results of argument extraction for the EFTE model. Figure (a) presents the results without the Event Boundary-Aware Encoding Module, while Figure (b) shows the results for the complete model. The pie chart's inner circle details the overall argument identification performance. The outer circle provides a statistical breakdown of the error proportion, specifically highlighting errors caused by arguments crossing event boundaries.

Table 8. Data Distribution Results for the Huawei Communication Fault Dataset

Data Distribution Details	Total data	Train data	Valid data	Test data
Total Data Volume	15000	11000	2000	2000
Number of Multi-Event Samples	11615	10110	724	781

5. Effectiveness Analysis

To evaluate the advanced performance of the EFTE method in communication fault analysis, we performed a comparative study against recent state-of-the-art techniques. (Note: Because of the confidentiality of the telecommunication dataset, we used a publicly available dataset from Huawei for this validation.)

5.1 Analysis of Boundary Discrimination Ability

To evaluate the effectiveness of the EFTE model in identifying event boundaries during event co-occurrence, we thoroughly analysed multi-event samples from two perspectives: (1) Argument Extraction Performance and (2) Internal Encoding Mechanism.

5.1.1 Analysis of Argument Extraction Performance

This subsection aims to demonstrate that the EFTE model can effectively identify the boundaries of different events by examining its superior argument extraction performance on multi-event samples. Specifically, EFTE is compared to high-performing methods like TagPrime and Scented, as well as

TabEAE and DEEIA, which are specifically designed to handle event co-occurrence. The results are shown in Figures 8 and 9.

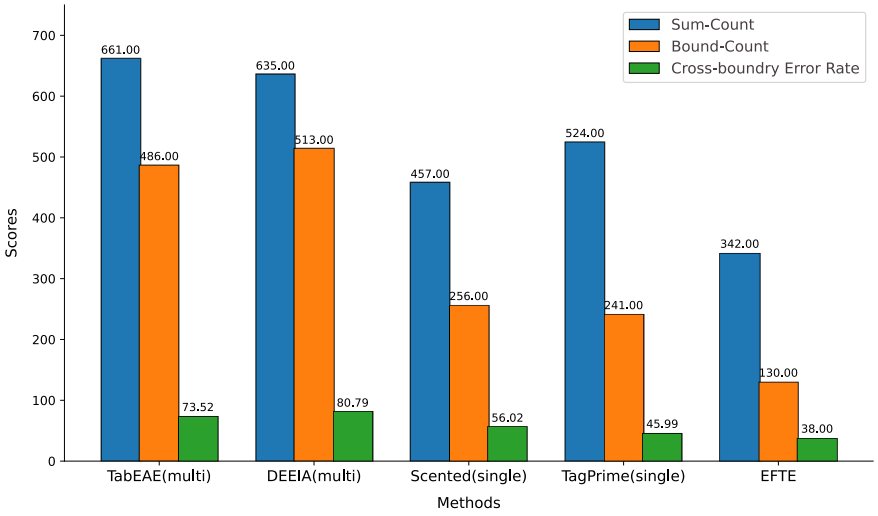

Figure 8. Statistical Chart of Argument Prediction Errors in Multi-Event Samples

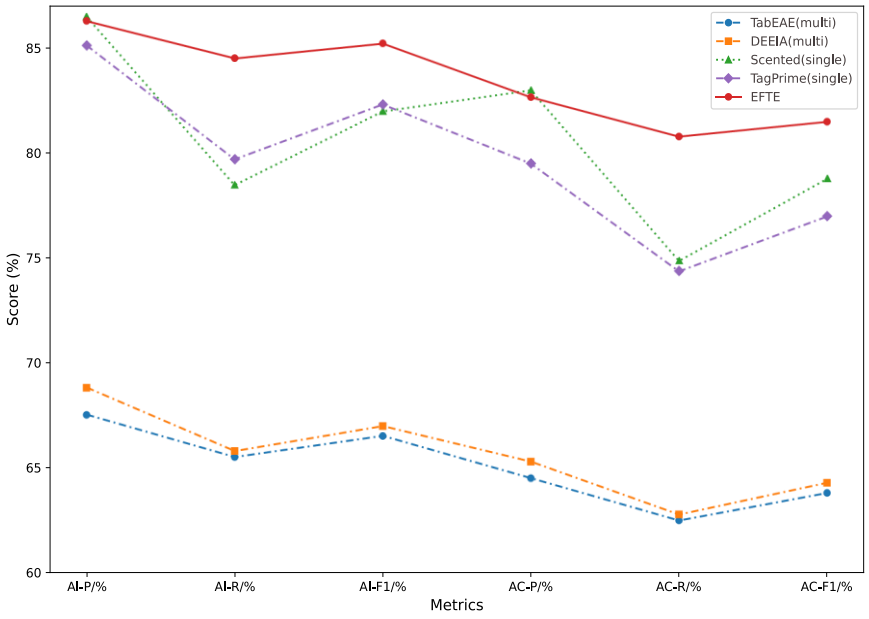

Figure 9. Statistical Chart of Argument Extraction Performance on Multi-Event Samples

Table 8 shows the statistics for the number of multi-event data samples in the dataset. It can be seen that multi-event samples make up 77.43% of the Huawei communication fault dataset, indicating that addressing the event co-occurrence problem in communication datasets is highly important and valuable. Figure 8 displays the distribution of argument prediction errors made by different methods on multi-event samples from the test set. In this figure, Sum-Count is the total number of argument prediction errors. Bound-Count is the number of instances within Sum-Count where the predicted argument content belongs to information from another event; that is, the expected argument failed to correctly identify the event boundary and crossed it. Su-Bo-Percent refers to the proportion of Bound-Count within Sum-Count. Likewise, Figure 9 shows the EAE performance of different methods on multi-event samples from the test

set.

Analyzing the results in Figure 8, it is clear that the number of argument prediction errors made by the TabEAE and DEEIA methods on multi-event samples is significantly higher than that of EFTE. In terms of the Su-Bo-Percent error ratio, the error rates for TabEAE and DEEIA are 35.31 and 42.78 percentage points greater than EFTE, respectively. Their error rates are also notably higher compared to the TagPrime and Scented baseline methods. This indicates that although TabEAE and DEEIA provide targeted solutions for the event co-occurrence problem and have achieved good results on datasets in related domains, they still have limitations in the communication domain—particularly in their ability to discriminate event boundaries, which needs improvement. This is because their approach to addressing event co-occurrence mainly involves associating prompt templates of different events and text information through attention mechanisms, using inter-event correlations to highlight event boundary information. This method of associating events via attention places high demands on prompt template design [27]; creating a high-quality prompt template requires considering many factors [28][29][30]. Additionally, because events are relatively independent and the communication domain often uses domain-specific terminology, overemphasizing inter-event correlations can sometimes blur the boundaries between events.

In contrast, our EFTE method mainly depends on the LLM to find specific events when solving multi-event problems. This improves event extraction while keeping event independence and requires fewer prompts for template creation, making it more transferable and reliable. As shown in the figures, the comparison of multi-event samples clearly shows that the EFTE approach has a strong edge in identifying event boundaries, effectively solving the event co-occurrence issue and performing superbly on the EAE task.

5.1.2 Analysis of the Internal Encoding Mechanism

To show that our proposed Boundary-aware Encoding Module can focus its attention on specific events during encoding, we performed a visual analysis of its internal mechanism. We chose the current state-of-the-art (SOTA) baseline model, Scented, for comparison. Using heatmaps, we visualized and compared the cross-attention distributions generated by the two models during argument extraction, as shown in Figure 10.

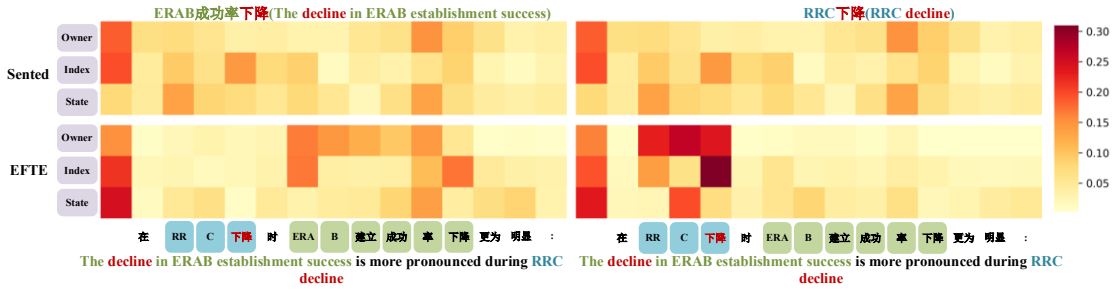

Figure 10. Visualizing the event boundary differentiation capabilities of the EFTE and Scented models via attention distributions. The left and right plots show how each model allocates attention differently across two events in a text, reflecting its capacity to distinguish between them.

Figure 10 uses the complex text, "The decline in ERAB establishment success is more pronounced during RRC decline", which contains two distinct events, as an illustrative example. We observe that when the Scented model processes this text, the cross-attention distributions for different argument roles (Owner, Index, State) exhibit a high degree of similarity when extracting different events. This indicates that the model fails to effectively distinguish the contextual information of the two separate events, leading to a confused attentional focus. This indicates that the model fails to effectively distinguish the contextual information of the two separate events, leading to a confusion of attention focus. This issue primarily arises because both events in this complex text belong to the same "Indicator Deterioration" category and share the same trigger word. Consequently, Scented cannot accurately locate event boundaries by relying solely on event type and trigger words, making it difficult to precisely focus on the context of the target argument's event.

In stark contrast, the EFTE model demonstrates precise boundary-aware capabilities. When extracting

the "The decline in ERAB establishment success" event, the argument roles adaptively focus their attention on relevant tokens such as "ERAB", "establishment", "success", and "decline". Similarly, when processing the "RRC decline" event, the attention accurately shifts and concentrates on the "RRC" and "decline" tokens. This result compellingly demonstrates that EFTE's boundary-aware module can dynamically allocate and isolate attention for different events, thereby significantly enhancing argument extraction performance in complex scenarios. This finding provides both a theoretical explanation and empirical support, at the level of the attention mechanism, for the superior performance exhibited by EFTE in handling multi-event samples.

The final experimental results show that EFTE, with its unique event boundary-aware mechanism, can better identify and differentiate the boundary features of various events, greatly enhancing the model's ability to handle complex multi-event samples.

5.2 Analysis of Domain-specific Terminology Recognition Ability

To evaluate the accuracy of the EFTE method in recognizing domain-specific terminology, the top-performing TagPrime and Scented methods from the comparative experiments were selected for analysis alongside EFTE specifically for this task. The results of domain-specific terminology recognition are shown in Table 9.

Table 9. Statistical Results of Domain-specific Terminology Recognition Performance on the Huawei Communication Fault Dataset

Model	EI-P	EI-R	EI-F1	EC-P	EC-R	EC-F1
TagPrime	78.04	78.79	78.42	77.27	78.01	77.64
Scented	81.23	82.33	81.77	80.10	81.11	80.60
EFTE	**83.24**	**84.10**	**83.67**	**82.48**	**83.34**	**82.91**

The experimental results in Table 9 show that the EFTE method outperforms both TagPrime and Scented. Specifically, for Term Identification metrics, EFTE achieved increases of 5.25 percent in Precision (EI-P) and 5.27 percent in Recall (EI-R). For Term Classification metrics, it showed improvements of 1.90 percent in Precision (EC-P) and 2.01 percent in Recall (EC-R). By attaining the best results across all performance metrics for domain-specific terminology prediction, the EFTE method confirms the effectiveness of its domain-specific terminology module in enhancing term recognition. Through the domain-specific terminology recognition module, the EFTE method improves the model's ability to identify terms in the communication domain and enhances overall model performance along with other modules.

5.3 Case Study

To better demonstrate the EFTE method's ability to distinguish event boundaries and accurately predict domain-specific terminology, EFTE is experimentally compared and analyzed against the more advanced TagPrime and Scented methods on a typical multi-event data example, as shown in Figure 11.

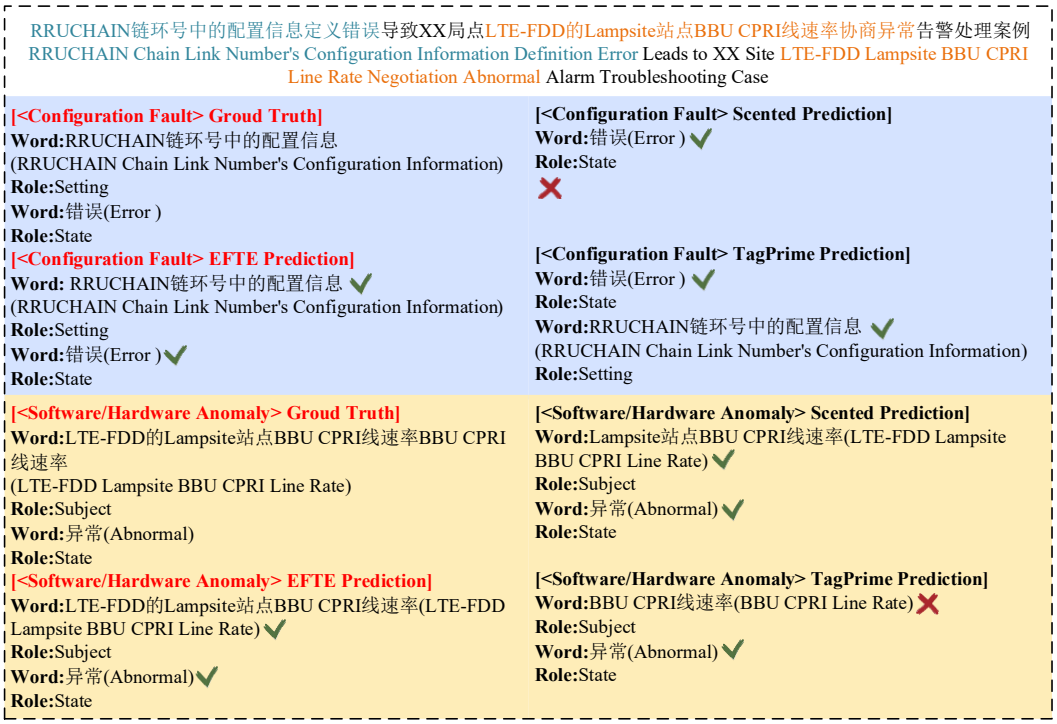

Figure 11. A comparison of argument extraction results from different models. The source text contains two distinct events, which are highlighted in yellow and blue, respectively.

Figure 11 shows the EAE results for the Scented, TagPrime, and EFTE methods on a text with multiple events. The yellow and blue sections represent the argument extraction results for events described by text of corresponding colours. Analysis reveals that the Scented method failed to extract the argument "RRUCHAIN Chain Link Number's Configuration Information" when extracting the event described by the blue text. While the TagPrime method successfully extracted this argument, it made an incomplete extraction error for the argument "LTE-FDD Lampsite BBU CPRI Line Rate" when extracting the blue event. This is mainly because these two methods, when processing multi-event samples, do not adequately focus on the model's ability for event discrimination and domain-specific terminology extraction, leading to some arguments being incorrectly extracted or not extracted at all. In contrast, EFTE, aided by its event-focusing prompt template and the domain-specific terminology prediction module, successfully and accurately extracted the event arguments, thereby enhancing the effectiveness of EAE.

The previous analysis shows that EFTE performs well on datasets that include multiple events and specialized communication domain terminology. Specifically, guided by the Event Boundary-aware Encoding Module, EFTE can thoroughly understand the boundary information between different events in the text. This enables it to filter out irrelevant information from other events during the extraction process for a particular event, focusing accurately on the semantic details of that event. Additionally, after securing a semantic representation focused on the specific event, EFTE, through its domain-specific terminology recognition module, improves its ability to extract domain-related terms within the event. It more precisely extracts arguments containing domain-specific terminology, resulting in strong overall performance in EAE.

6. Conclusion

This paper proposes an event-focused and terminology-enhanced EAE method for communication faults, based on LLM, to address the challenges of event co-occurrence and the difficulty in recognizing domain-specific terminology in communication domain data. To handle event co-occurrence, unlike current methods that use attention mechanisms to extract all event arguments simultaneously, this work leverages an LLM to design an Event Boundary-aware Encoding Module. This module guides the model to focus on specific event content during encoding, improving its ability to distinguish event boundaries.

Additionally, to enhance recognition of domain-specific terminology, a Semantically-enhanced Domain-specific Term Recognition Module was proposed, also using an LLM, which increased the accuracy of identifying arguments related to communication terms. To validate the method's effectiveness, experiments were conducted on the publicly available Huawei communication fault dataset and a constructed telecommunication fault dataset. The results demonstrate that the proposed method effectively addresses the issues of event co-occurrence and the challenge of recognizing domain-specific terminology. Finally, the effectiveness analysis further confirms the efficacy of the EFTE method for the EAE task.

It is worth noting that in the fine-tuning of the LLM, this study did not perform a detailed analysis of the performance improvements achievable through different prompt engineering techniques. Additionally, fine-tuning requires certain computational resources. Therefore, future research should explore other prompt engineering approaches and consider how to modify the model to reduce computational demands, further improving the accuracy and robustness of the extraction task.

Data Availability Statement

Our experimental evaluation utilizes two primary datasets: (1) the publicly available Huawei Communication Fault Dataset, accessible through the GitHub repository (https://github.com/shenzaimin/CCKS-2021-Huawei-Event-Extraction/tree/main/data/train_fix), and (2) a proprietary Telecommunication Fault Dataset that cannot be made publicly available due to confidentiality agreements and privacy considerations.

Author Contributions

Le Zhang: Proposed research ideas, reviewed and revised the paper;

Yangke Xu: Designed experiments based on the research ideas and wrote the paper;

Leihan Zhang: Improved the algorithm ideas and provided suggestions for revising the paper;

Acknowledgements

This research was funded by: (i) The National Natural Science Foundation of China, grant number 72501034. (ii) The General Project of the Science and Technology Program of the Beijing Municipal Commission of Education, grant number KM202311232001. (iii) The National Natural Science Foundation of China, grant number 62102044. (iiii) The National Key R\&D Program of China, grant number 2018YFB1004100.

References

[1] Wang H, Zhu T, Wang M, et al. A prior information enhanced extraction framework for document-level financial event extraction[J]. Data Intelligence, 2021, 3(3): 460-476.

[2] Simon É, Olsen H, You H, et al. Generative Approaches to Event Extraction: Survey and Outlook[C]//Proceedings of the Workshop on the Future of Event Detection (FuturED). 2024: 73-86.

[3] Liu P, Yuan W, Fu J, et al. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing[J]. ACM computing surveys, 2023, 55(9): 1-35.

[4] Peng J, Yang W, Wei F, et al. Prompt for extraction: Multiple templates choice model for event extraction[J]. Knowledge-based systems, 2024, 289: 111544.

[5] Liu W, Zhou L, Zeng D, et al. Beyond Single-Event Extraction: Towards Efficient Document-Level Multi-Event Argument Extraction[C]//Findings of the Association for Computational Linguistics ACL 2024. 2024: 9470-9487.

[6] Ma Y, Wang Z, Cao Y, et al. Prompt for Extraction? PAIE: Prompting Argument Interaction for Event Argument Extraction[C]//Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2022: 6759-6774.

[7] I-Hung Hsu, Kuan-Hao Huang, Shuning Zhang, Wenxin Cheng, Prem Natarajan, Kai-Wei Chang, and Nanyun Peng. 2023a. TAGPRIME: A unified framework for relational structure extraction. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL).

[8] Xue Jiwei, Hu Xinyuan, Xue Pengjie. Research on Document Level Event Argument Extraction Method Based on Prompt Learning[J]. Computer Technology and Development,2024,34(06):125-131.

[9] Yang Y, Guo J, Shuang K, et al. Scented-EAE: Stage-Customized Entity Type Embedding for Event Argument Extraction[C]//Findings of the Associationfor Computational Linguistics ACL 2024. 2024: 5222-5235.

[10] Li S, Ji H, Han J. Document-Level Event Argument Extraction by Conditional Generation[C]//2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021. Association for Computational Linguistics (ACL), 2021: 894-908.

[11] Huang K H, Hsu I H, Natarajan P, et al. Multilingual Generative Language Models for Zero-Shot Cross-Lingual Event Argument Extraction[C]//Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2022: 4633-4646.

[12] Hsu I H, Huang K H, Boschee E, et al. DEGREE: A Data-Efficient Generation-Based Event Extraction Model[C]//Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2022: 1890-1908.

[13] Xu Z, Wang P, Ke W, et al. Incorporating schema-aware description into document-level event extraction[C]//Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence. 2024: 6597-6605.

[14] He Y, Hu J, Tang B. Revisiting Event Argument Extraction: Can EAE Models Learn Better When Being Aware of Event Co-occurrences?[C]//Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2023: 12542-12556.

[15] Peng J, Yang W, Wei F, et al. Event co-occurrences for prompt-based generative event argument extraction[J]. Scientific Reports, 2024, 14(1): 31377.

[16] Wang L, Ma C, Feng X, et al. A survey on large language model based autonomous agents[J]. Frontiers of Computer Science, 2024, 18(6): 186345.

[17] Ma Y, Cao Y, Hong Y C, et al. Large Language Model Is Not a Good Few-shot Information Extractor, but a Good Reranker for Hard Samples![C]//The 2023 Conference on Empirical Methods in Natural Language Processing.

[18] Zhao S ,Zhou T ,Jin Z , et al.AWeCita: Generating Answer with Appropriate and Well-grained Citations Using LLMs[J].Data Intelligence,2024,6(04):1134-1157.

[19] Vincent, Qian Li, Hu Maodi, et al. Review of research progress on question answering technology based on large language models [J/OL]. Data Analysis and Knowledge Discovery, 1-17 [2024-06-14].)

[20] Hu E J, Shen Y, Wallis P, et al. Lora: Low-rank adaptation of large language models[J]. ICLR, 2022, 1(2): 3.

[21] Wei J, Wang X, Schuurmans D, et al. Chain-of-thought prompting elicits reasoning in large language models[J]. Advances in neural information processing systems, 2022, 35: 24824-24837.

[22] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. arXiv preprint arXiv:2412.15115, 2024b.

[23] Xue L, Constant N, Roberts A, et al. mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer[C]//Proceedings of the 2021 Conference of the North American Chapter of the Association

for Computational Linguistics: Human Language Technologies. Association for Computational Linguistics, 2021.
[24] Lafferty J, McCallum A, Pereira F. Conditional random fields: Probabilistic models for segmenting and labeling sequence data[C]//Icml. 2001, 1(2): 3.
[25] Chen Y, Xu L, Liu K, et al. Event extraction via dynamic multi-pooling convolutional neural networks[C]//Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). 2015: 167-176.
[26] Du X, Cardie C. Event Extraction by Answering (Almost) Natural Questions[C]//Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). 2020: 671-683.
[27] Liang C. Event Argument Extraction with Enriched Prompts[J]. arxiv preprint arxiv:2501.06825, 2025.
[28] Xiang W, Zhan C, Wang B. Daprompt: Deterministic assumption prompt learning for event causality identification[J]. arXiv preprint arXiv:2307.09813, 2023.
[29] Li F, Peng W, Chen Y, et al. Event extraction as multi-turn question answering[C]//Findings of the Association for Computational Linguistics: EMNLP 2020. 2020: 829-838.
[30] Zhong Z, Chen D. A Frustratingly Easy Approach for Entity and Relation Extraction[C]//2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021. Association for Computational Linguistics (ACL), 2021: 50-61.

Author Biography

Le Zhang is an Associate Professor at the School of Computer Science, Beijing Information Science and Technology University. She obtained her doctoral degree from Beijing University of Posts and Telecommunications. Her research focuses on natural language processing, large language models, knowledge graphs, and multimodal information processing. She is currently working on a communication fault diagnosis platform based on large language models and knowledge graphs, with some functions already successfully deployed in a telecom company.

Yangke Xu is currently pursuing a Master's degree at Beijing Information Science and Technology University. His research interests primarily lie in Natural Language Processing (NLP), with a focus on areas such as multimodal summarization and information extraction.

Leihan Zhang is an Associate Professor at the School of Economics and Management, Beijing University of Posts and Telecommunications. He holds a Ph.D. in Computer Science and Technology and completed his postdoctoral research at the Wangxuan Institute of Computer Technology, Peking University. His research focuses on social computing, understanding and generation of internet memes, and large model-based intelligent agents. He is currently dedicated to developing a platform for sharing and creating image-and-text-based internet memes, as well as a knowledge graph-based platform for representing AI risk events.

A: Experimental Environment Parameter Settings

As the experiments involve both the fine-tuning of the LLM and the training of the EAE model, two distinct experimental environments are utilized. During the fine-tuning of the LLM, the batch size for fine-tuning training is set to 4, and the model is trained for a total of 5 epochs. The fine-tuning experiments are conducted on a server with the Ubuntu 20.04 operating system. Specific details of this experimental environment are provided in Table 10.

Table 10. Experimental Environment Information (LLM)

Operating System	Linux
CPU	Intel(R) Xeon(R) Gold 6226R CPU @ 2.90GHz
GPU	NVIDIA-SMI RTX A6000
Python	3.11.0
Pytorch	2.3.0
CUDA	12.2

During the experiments for training the EAE model, 50 epochs are run. After each epoch, the performance on the test and validation sets, along with the current model state, is saved. The model that produces the best decoding results is chosen as the outcome of the experiment. The EAE experiments are performed on a server with Ubuntu 20.04 as the operating system. Specific details of this experimental environment are provided in Table 11.

Table 11. Experimental Environment Information (EFTE)

Operating System	Linux
CPU	Intel(R) Xeon(R) Gold 5115 CPU @ 2.40GHz
GPU	Tesla V100-PCIE-16GB
Python	3.8.0
Pytorch	2.1.2
CUDA	12.3