SMALLM: a Local Small Model Augmented a Cloud-based Large Language Model for Chinese Named Entity Recognition in Low-resource Industries

(Preprints, will be extensively revised for further publication)

Abstract

Named Entity Recognition (NER), a pivotal task in natural language processing (NLP) aiming to extract valuable entities from unstructured texts, is instrumental in various applications, including information retrieval, compliance checking, and question answering. Despite its success in processing general data, implementing NER in specialized industries remains challenging due to inadequate labeled data, computational constraints, and linguistic complexities. While deep learning (DL) models are robust and popular for NER, including cutting-edge large language models (LLMs), they often demand extensive, high-quality data and computational power. Therefore, this study introduces "GPTBERT," a novel NER method that seamlessly fuses the state-of-the-art LLM, GPT-3.5 (Generative Pre-Trained Transformer version 3.5), with a local model, BERT (Bidirectional Encoder Representation from Transformers). Compared to BERT, GPTBERT reduces the computational demands and realizes fast training and less reliance on data, thereby making NER more accurate, affordable, and adaptable across different industries, such as clinics, construction, and the military. GPTBERT only requires updating <0.2% of the entire model parameters to reach or surpass the performance of full fine-tuning; it also provides a unique solution to the constraints posed by closed source models like GPT-3.5 via converting their responses into one-hot vectors for efficient integration. In industries, GPTBERT delivers an effective, cost-efficient, and adaptable solution for various applications, including compliance checking in the construction industry, epidemiological studies in the clinic domain, strategic analysis in the military sector, etc.

1 Introduction

Named Entity Recognition (NER), one of the natural language processing (NLP) tasks, aims to identify entity boundaries and predefined types (e.g., "Address" and "People") from textual sentences [1]. NER is a long-standing task in artificial intelligence (AI). Various NER methods, including rule-based methods, deep learning models (DL), and state-of-the-art (SOTA) large language models (LLMs), have been proposed and applied [2]. NER is a cornerstone of many AI-based applications, such as information retrieval, question answering [3], and compliance checking [4]. The applications substantially facilitate production and management activities in industries that involve intensive text-handling tasks, such as manufacturing [5], construction [6], military [7], and clinics [8]. Current NER methods have reached high

performance in general datasets, such as CoNLL03 and OntoNotes5.0 [9]. However, applying NER methods in specific industries is still challenging due to the limited coverage of rules and features, inadequate labeled data, and constrained computational resources. In addition, compared to languages with clear segmentation between words, such as English, languages without spaces in sentences, e.g., Chinese, present greater challenges on domain-specific NER because the difficulty in determining entity boundaries and complex grammars can further increase pre-processing workload and the demands of data and model capability [10].

In industry-level NER applications, DL models, such as the long-short term memory (LSTM) [11], Gated Recurrent Unit (GRU) [12], and Transformer [13] and its variants [14-16] are increasingly used compared to rule-based and machine learning (ML) methods. This is because DL models exhibit strong learning capabilities, contextual understanding, and superior adaptability data variances thereby reducing reliance on manual labor [17]. There are two perspectives when adapting DL models to industrial NER: regarding it as either a sequence labeling problem or a generation problem.

Models belonging to the sequence labeling group follow the "pre-training and fine-tuning" procedure. Generally, these models consist of three main components: 1) a pre-trained language model (PLM) as the backbone mapping discrete language symbols (e.g., characters or words) to embeddings (i.e., numerical vectors) that encode semantic meanings of the symbols, 2) additional task-specific layers sequentially decoding the embeddings to entity types in certain domains, and 3) domain datasets for model training and validation. The latter two steps are called "fine-tuning." However, whenever the industry changes, both the datasets and task-specific layers need to be reconstructed or modified [17]. More importantly, to achieve satisfactory NER performance and adaption to downstream tasks, it is necessary to fine-tune all parameters (often millions or even billions) in both the backbone model and added layers [16].

As for the text generation approaches, Large Language Models (LLMs) are popular options. LLMs adopt a unified architecture to address all tasks, showcasing exceptional capabilities in few-shot and zero-shot learning across various NLP tasks compared to traditional DL models. Nevertheless, LLMs perform worse than sequence labeling models specifically trained on domain datasets [18]. This phenomenon is also evident in the cutting-edge generation models like GPT-3.5 and GPT-4 developed by the Open-AI (more details are presented in Section 5.4.1). Moreover, in commercial and closed source models such as GPT-3.5 and GPT-4, users can only access plain texts instead of numerical outputs or

intermediate embeddings when using the API, which hinders usage of the model outputs and further inter-model optimization.

In addition, both solutions suffer from excessive requirements for high-quality data and computing resources. For one thing, it is labor-intensive and time-consuming to define named entity types, establish annotation rules, and annotate and update enough training data for either PLMs or LLMs. LLMs, such as FLAN-T5, can require additional fine-tuning using specialized prompts to boost performance [19]. For another, computing resources are commonly constrained in practice. Usually, only general-purpose CPUs or low-performance GPUs are available, which do not allow luxury training or fine-tuning [20]. Therefore, we propose a novel NER method, namely, GPTBERT. The core lies in incorporating prior knowledge from the SOTA GPT-3.5 into a local model through a fusion layer, which can make NER applicable and affordable across different industries, even on small samples, without compromising performance. In this study, the effectiveness of GPTBERT was validated on datasets from three different industries: construction, clinic, and military. Our contribution can be summarized as follows:

1) Efficient and light-weight training that can adapt to datasets in different industries with different scales and data distributions: GPTBERT achieves or surpasses traditional complete fine-tuning methods of local small models by updating less than 0.2% of its total parameters. As fine-tuning samples increase, GPTBERT consistently outperforms GPT-3.5, which lacks domain-specific tuning. Direct fine-tuning LLMs demands significant computing power and sophisticated design of instructions and prompts. In contrast, GPTBERT introduces a novel fine-tuning approach. It integrates a small local model with the LLM, which outperforms both the local and LLM by only fine-tuning the small model. Thus, compared to traditional fine-tuning methods of local small model and LLM, GPTBERT substantially reduces computational resource needs. This light-weight model can be trained on standard CPUs and adapted across various industries.
2) Less data reliance: GPTBERT incorporates both BERT and GPT predictions. This leverages the in-context zero-shot learning capability of LLMs. Therefore, in clinic and military datasets, GPTBERT can outperform GPT-3.5 under conditions where the training dataset is extremely limited (<=50 samples), while classical methods fail.
3) Convenient implementation: To overcome the closed-source constraint when using commercial LLMs, text responses returned from GPT-3.5 API are converted into one-hot vectors, fed into

GPTBERT, and then processed by the fusion layer. As such, GPTBERT can effectively integrate the predictions from GPT-3.5 with the text embeddings from the frozen pre-trained BERT.

2 Related Work

2.1 Deep Learning Models for NER

In recent years, DL-based NER models become the dominant option and achieve SOTA results. Compared to traditional rule-based approaches and ML models requiring manually crafted features, DL is beneficial in terms of automatically discovering hidden features, thereby saving time for manually crafting rules and features. Supposing the domain dataset for NER tasks has been established, there are two main concerns of developing DL-based NER models, namely, text embeddings and additional task-specific layers.

2.1.1 Underlying Principles of DL-based NER

Text embeddings are numerical representations derived from text and are typically used to capture semantic meaning. Text embeddings can be categorized as character-level, subword-level and word-level based on granularities. According to whether text embeddings dynamically change based on the context, they can be divided into static embeddings and dynamic embeddings. Static embeddings are trained and frozen as lookup matrices using shallow models, such as NNLM [21], Word2vec [22], FastText [23], and Glove [24]. While training them is easy and cost-effective, they have two main drawbacks. First, they don't capture the context well, especially for words with multiple meanings in different situations. Second, they struggle with out-of-vocabulary words, as the model can only recognize words it encountered during training. In Chinese NER tasks, the out-of-vocabulary drawback of static embeddings becomes more prominent due to the absence of clear word boundaries in Chinese. Incorrect word segmentation can lead to a word not being in the vocabulary, resulting in the absence of its embedding.

On the other hand, dynamic embeddings generated by PLMs such as ELMo [14], GPT [15], BERT [16], ERNIE [25], ALBERT [26], and NEZHA [27] have been developed. They can address the limitations of static embeddings by generating context-dependent vectors of each token (i.e., a text symbol). The token can be a Chinese character, a subword, or a word. Most of the PLMs adopt a Transformer-based architecture [28]. The Transformer employs an encoder-decoder structure featuring multiple stacked self-attention modules, which can effectively model the long-range contextual dependencies in sentences. There are two steps in training a PLM: pretraining and fine-tuning. Pretraining is the initial training phase

where a model is trained on a large corpus of text through unsupervised or semi-supervised tasks (e.g., masked tokens prediction) to learn general language representations and patterns. Dynamic embeddings generated from this stage encapsulate a generalized understanding of language but are not specialized in particular tasks. Fine-tuning is the subsequent phase where all the parameters of the pretrained model are further fine-tuned on domain-specific or task-oriented data, optimizing the dynamic embeddings for particular domains. However, as implementing the fine-tuning step is not always imperative, the dynamic embeddings derived from pre-training can also be directly deployed for downstream tasks. This technique is known as the feature-based approach. The feature-based approach is more computationally efficient, although it may not realize as high performance as full fine-tuning in certain tasks [16]. Many studies have demonstrated that dynamic embeddings lead to considerable NER performance enhancements. Among these embeddings, BERT, which leverages the attention mechanism and jointly condition on both the left and right context in all model attention layers, is the most widely used one in NER tasks [29-31].

Adding task-specific layers is the next stage of developing a domain-specific NER model. The layer takes text embeddings as input and generates a sequence of tags, where the tags are pre-defined using task-specific data to train the model. Two primary implementation methods exist: Multi-Layer Perceptron (MLP) with softmax and MLP with Conditional Random Field (CRF). MLP incorporates multiple fully connected layers which are directly used to obtain the score of each label corresponding to the token. The limitation of using MLP with softmax is that each label is independently predicted, regardless of its surrounding tokens. Thus, CRF is widely applied as the supplementary tag decoder, which serves as a global conditioned random field on the observation sequence and is good at capturing tag transition dependencies [32]. Currently, NER models with CRF can achieve SOTA performance on CoNLL03 and OntoNotes5.0 [9].

2.1.2 Industry-level Applications of DL-based NER

DL-based NER models have been applied in NER tasks across different industries, including but not limited to architecture, engineering, construction, transportation, clinics, and the military. In the early stages, static embeddings were the popular option. Moon et al. [6] combined the Word2Vec embeddings and Bi-LSTM model, which achieved a satisfactory NER performance of 0.919 precision and 0.914 recall when extracting construction specification samples. Lu et al. [33] trained the Bi-LSTM+CRF model with Word2Vec embeddings, achieving an F1 score of 73.9% in terms of extracting military entities on their

custom dataset. Martinez Soriano et al. [34] introduced a clinic concept recognition system where static embeddings were generated by an unsupervised model from a vast clinical report database. The model demonstrated high accuracy in clinic concept identification and realizing enhanced document search. However, the studies require creating or updating the domain-specific vocabulary if the task or data are modified. Otherwise, there will be out-of-vocabulary data and damage the NER performance. Moreover, each time the domain changes, the embeddings are trained from scratch using a new dataset, revealing the poor generalization capabilities of the methods. As such, NER models enabled by dynamic embeddings has become the mainstream method recently. For instance, Song et al. [35] trained a BERT on a corpus of 1,000 annotated construction safety accident reports, significantly outperforming traditional models in recognizing entities within construction safety accident texts. Zheng et al. [36] fine-tuned several PLMs on construction domain corpora to extract project entities such as equipment, tasks, and materials. Vunikili et al. [37] explored the usage of BERT trained on general domain Spanish texts to extract tumor morphology from clinical reports. Lu et al. [38] integrated BERT with Bi-LSTM-CRF to extract specialized military entities and reached better performance than solely using the sequential model.

Despite the achievements, using the above DL models to solve real-world NER problems is still impractical. DL models need to fine-tune all parameters on domain datasets to yield satisfactory performance [35-38]. However, the number of parameters easily exceeds one hundred million [14-16, 25-27], and fine-tuning requires substantial domain-specific labeled data and computational resources. The feature-based approach is more computationally efficient and does not involve updating the parameters of the entire pre-trained model. However, the NER performance is often compromised using feature-based approaches, especially when the domain significantly differs from the one where the model was pre-trained [16].

2.2 Large Language Models for NER

LLMs have demonstrated remarkable capabilities in various NLP tasks [39-42]. LLMs, typically constructed upon the Transformers architecture, differ from PLMs by embracing a unified text-to-text generation framework, allowing them to adapt to various tasks without changing the model structure. The number of parameters of LLMs has been continually increasing with advancements in technologies. The cutting-edge LLM, GPT-4, has more than a trillion parameters and shows human-level performance on various professional and academic benchmarks [43].

Wei et al. observed that a model with a sufficiently large number of parameters has emergent abilities that do not present in smaller models, namely, the LLMs can realize effective few-shot or zero-shot learning with straightforward human instructions (called prompts). Brown et al. [39] proposed few-shot prompting to leverage emergent abilities, which is also named in-context learning (ICL). The method provides a prompt with a few demonstrations of the unseen task to the LLM, enabling the model to perform new tasks without further training or updating parameters. The emerging capabilities of LLMs contribute to their significant advancement in task-agnostic and few-shot performance, occasionally even rivaling prior SOTA fine-tuning approaches [44]. Several studies have verified that the effectiveness of ICL is highly affected by the design of prompts [45-47]. To construct an effective prompt, it is necessary to select appropriate demonstrations [48], arrange them in a coherent sequence [49], and define a suitable format, including adding a description of the task [19].

Due to the remarkable performance and generalization capacity of LLMs, they can adapt to a wide range of tasks, including NER. Wang et al. proposed GPT-NER, which defined the output format in the prompt and employed a self-verification strategy by prompting LLMs to verify whether the extracted entities belong to a labeled entity tag. GPT-NER demonstrated comparable results to fully supervised baselines and performed well in low-resource and few-shot scenarios on general-domain datasets, including CoNLL2003 and OntoNotes5 [50]. On the other hand, Han et al. evaluated the performance of ChatGPT on 17 datasets with 14 information extraction tasks under the zero-shot, few-shot, and chain-of-thought scenarios, and they found a considerable performance gap between ChatGPT and SOTA results [18]. Specifically, it is necessary to fine-tune ChatGPT using vast data and prompts taking specialized forms (e.g., chain-of-thought) so that it can excel at NER tasks in certain domains [19]. However, given the excessively large parameter size in LLMs, fine-tuning is unaffordable. Additionally, current commercial LLMs, e.g., ChatGPT and GPT-4, can only be accessed by calling their APIs. The model parameters and numerical results are not disclosed to the public, which limits their flexibility in downstream tasks. For open-source LLMs, like Llama [51] and Alpaca [52], even though their model parameters are accessible, the limited parameter scale restricts their capability to rival ChatGPT in many tasks, including NER.

In summary, despite advancements in PLM and LLM-based approaches, developing effective NER tools suitable for features and needs in specific industries is still challenging. Existing PLMs suffer from

the lack of computing resources and annotated datasets. The scarcity of annotated datasets forces the models to compromise performance when adapting NER tasks to industries. As for LLMs, e.g., ChatGPT and GPT-4, they can achieve plug-and-play functionality in NER due to their excellent generalization capabilities. However, their applications are less flexible than PLMs owing to API constraints. Besides, without further tuning, LLMs perform significantly worse than SOTA PLMs. To our knowledge, there is no effective approach that can leverage the generalization ability of LLMs and the precise NER capacity of PLMs. Thus, we propose the GPTBERT to enhance NER performance in specific domains. The GPTBERT fuses the prior knowledge provided by GPT-3.5 with a local BERT model. As introduced in the experiments, by only using a small amount of data and adjusting a minimal number of parameters, GPTBERT can adapt to NER tasks in different domains and outperform the feature-based approach with the same model architecture but ignoring GPT-3.5 predictions.

3 GPTBERT Framework

As shown in Figure 3-1, GPTBERT consists of four modules: data pre-preprocessing module, GPT-3.5 prediction module, dynamic embedding module, and fused classification module. Particularly, the data pre-preprocessing module tokenizes and labels raw text sentences. In the GPT-3.5 prediction module, a NER-specific prompt is developed with few-shot demonstrative examples, and the specialized Open-AI API is invoked to ask GPT-3.5 to predict named entities from input sentences. In the semantic encoding module, the input tokens are transformed into semantic embeddings using the pre-trained BERT model. In the fused classification module, both GPT-3.5 predictions and semantic embedding are fused, and then a CRF layer is employed to produce the final predictions.

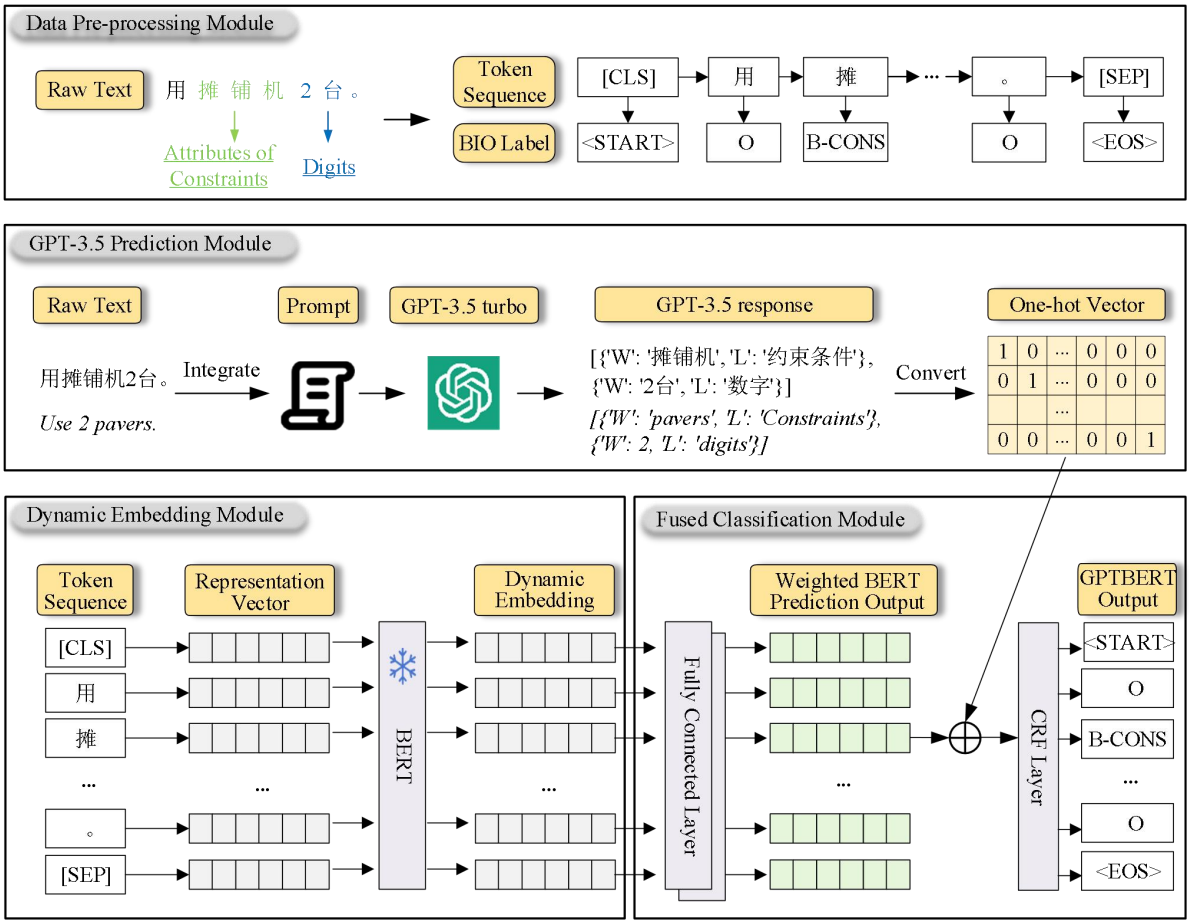

Figure 3-1 Main pipeline of GPTBERT.

3.1 Data pre-preprocessing module

The data pre-processing includes two steps. First, raw texts are tokenized into meaningful pieces. Second, entity labels of these texts are transferred into a BIO format. In this paper, a "span" denotes a contiguous text with arbitrary length. An "input sequence" refers to a text sequence that contains one or more sentences. The "entity type" indicates the category of entities of interest in certain domains, excluding the "Other" type to tag all texts falling out of the pre-defined entity types.

The study utilizes the built-in BERT tokenizer to split each Chinese raw text into Chinese characters (i.e., tokens) that are pre-defined in the token vocabulary. Except for Chinese characters, several special tokens are added e.g., [CLS], [SEP]. The [CLS] token is a special classification token placed at the beginning of a sequence, and its embedding serves as the aggregated semantic representation of the entire sequence for classification tasks. The [SEP] token acts as a separator, delineating different sentences (if any) within the input sequence. Therefore, the raw text is tokenized into a token sequence $X = (x_1, \ldots, x_n)$. Subsequently, these tokenized texts are labeled according to the BIO format, where "B" denotes the beginning token of a span corresponding to an entity, "I" represents all the characters that a span covers,

including the ending token, and "O" stands for tokens that do not belong to any entity types, i.e., the "Other" type. An example is shown in Figure 3-2.

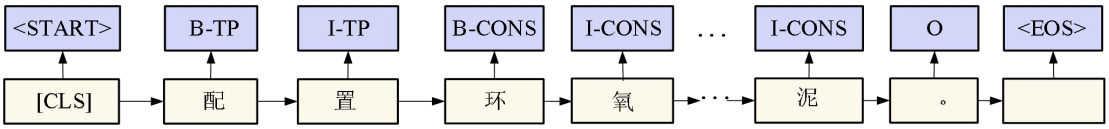

Figure 3-2 An example of BIO labeling. The full sentence shown in this figure is "配置环氧树脂胶泥" means "equipped with epoxy resin cement".

3.2 Dynamic embedding module

The GPTBERT uses BERT as the local model to comprehend the semantic meanings of text tokens and generate dense embeddings for downstream NER. BERT is preferred over other models for NER tasks since it captures contextual information effectively through pre-training on large and general text corpora, allowing it to understand the surrounding context and contextually identify entities, resulting in superior NER performance compared to traditional models [53]. BERT is pre-trained on a vast corpus of unlabeled data of the general world knowledge through two unsupervised approaches: masked language modeling (MLM) and next sentence prediction (NSP). MLM involves randomly masking words in a sentence and training the model to predict the masked words based on their context. NSP aims to determine whether two sentences are contiguous or not, helping the model understand sentence relationships and coherence. The pre-training approaches enable BERT to capture rich contextual information and learn contextualized word embeddings. The GPTBERT adopts the feature-based approach. As such, we froze the parameters of the pre-trained BERT and directly used them in our domain-specific NER tasks. Compared to full-parameter fine-tuning, the feature-based approach is more computationally efficient while can sacrifice performance [16]. Thus, we develop a feature fusion layer (the details are introduced in section 3.4), which enhances the performance of the feature-based approach on the specific-domain dataset by incorporating prior knowledge from the SOTA generation model GPT-3.5.

3.2.1 Input representation generator

The input representation generator is responsible for transforming a token sequence $\mathrm{X} = (\mathrm{x}_1, \ldots, \mathrm{x}_\mathrm{n})$ into a sequence of representation vector $R = (r_1, r_2 ... r_n)$ as the input to BERT. Each representation vector r_i is constructed by summing three types of embeddings: token embeddings, segment embeddings, and position embeddings. Token embeddings are obtained by looking up the embedding matrix using the indices of the tokens in the vocabulary. This embedding matrix is randomly initialized and subsequently trained during the pre-training phase. Segment embeddings are used to distinguish different sentences.

Position embeddings encode the position of each token in the input sequence, thereby enabling the model to understand the context of the current token and the holistic order of the sequence.

3.2.2 Dynamic embedding encoder

The dynamic embedding encoder of BERT encodes the input sequence of representation vector $R = (r_1, r_2 ... r_n)$ into dynamic embeddings $E_{BERT} = (e_1, \ldots, e_n)$. The model is composed of a stack of multiple transformer blocks with identical structures. Each block consists of two sub-layers: the first is a multi-head self-attention layer, and the second is a fully-connected feed-forward layer. Following each sub-layer, a residual connection with layer normalization, namely the normalization layer, is employed. Figure 3-3 shows the architecture of the dynamic embedding encoder.

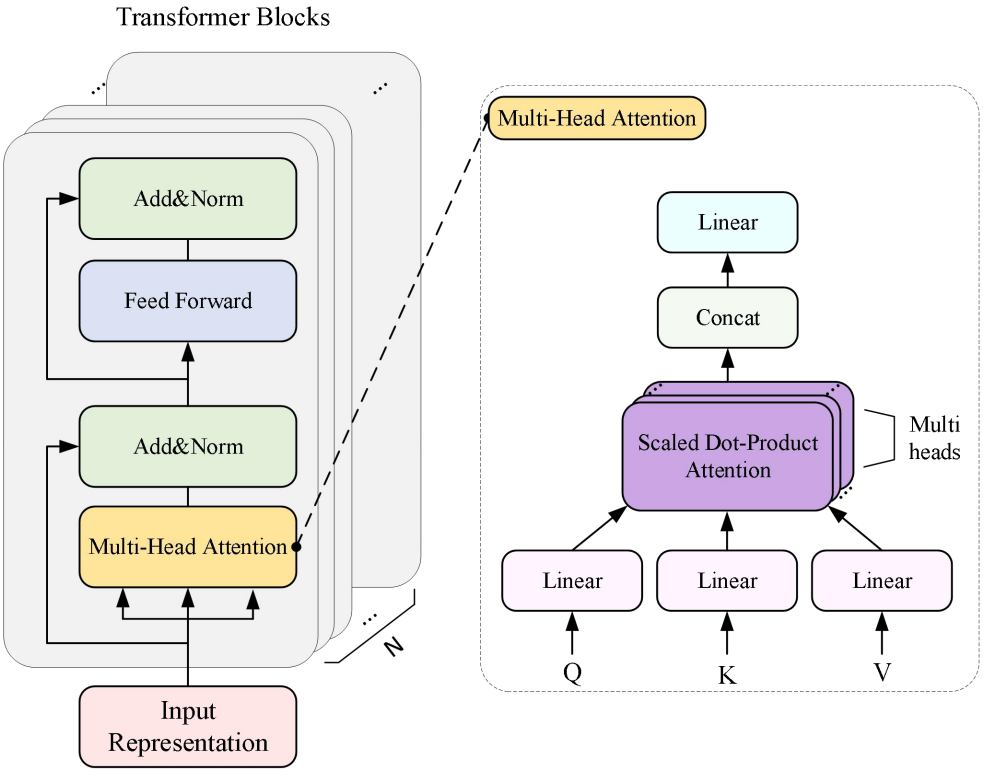

Figure 3-3 Architecture of dynamic embedding encoder

Multi-head self-attention layer: The multi-head self-attention layer is the core of the BERT structure. Through multiple attention heads, an input representation vector r_i is mapped into multiple representation sub-spaces. Specifically, for each attention head, the input R is transformed into Q, K, V by multiplying the matrix W^Q, W^K, W^V (Eq.1) and then producing the attention value $Head_i$ by processing the self-attention function (Eq. 2). In other words, the attention value is the weighted sum of value V_i, where the weight is calculated by query Q with related K. Then, the output of the multi-head self-attention layer MultiHead is obtained by applying a linear transformation to the concatenation of multiple $Head_i$ values (Eq. 3), where

$i = 1,2, ...K$ and K is the number of heads.

$$\mathrm{Q} = \mathrm{RW}^{\mathrm{Q}}, \mathrm{K} = \mathrm{RW}^{\mathrm{K}}, \mathrm{V} = \mathrm{RW}^{\mathrm{V}} \quad (1)$$

$$\mathrm{Head_i}(\mathrm{Q}, \mathrm{K}, \mathrm{V}) = \mathrm{softmax}(\frac{\mathrm{QK^T}}{\sqrt{\mathrm{d_k}}})\mathrm{V} \quad (2)$$

$$\mathrm{MultiHead} = \mathrm{Concat}(\mathrm{Head}_1, \ldots, \mathrm{Head_i})\ \mathrm{W^o} + \mathrm{b^o} \quad (3)$$

Fully-connected feed-forward layer: The output, i.e., MultiHead is processed by a feed-forward network containing 2 fully-connected feed-forward layers with a GeLU activation function.

$$\mathrm{FFN(MultiHead)} = \mathrm{GeLU(MultiHead}\ \mathrm{W}_1 + \mathrm{b}_1)\mathrm{W}_2 + \mathrm{b}_2 \quad (4)$$

Normalization layer: The normalization layer (Eq. 5) is added after each sub-layer: multi-head self-attention layer and the fully-connected feed-forward. The step "add" is to sum the input I of a sub-layer with its output, avoiding vanishing gradient problem. Following step "norm" is normalization, which helps to stabilize the gradient descent step, and lets models converge faster for a given learning rate.

$$\mathrm{Output_{sublayer}} = \mathrm{LayerNorm(I + SubLayer(I))} \quad (5)$$

In this method, BERT are initialized with the pre-trained weight from "BERT-base-chinese" which is published on hugging face and pre-trained for Chinese. "BERT-base-chinese" adopts 12 transformer encoder blocks and 12 attention heads. In "BERT-base-chinese", each Chinese character, namely each token, is firstly transformed into input representation and then processed by 12 transformer encoder blocks each of which has 12 attention heads. After that the dynamic embedding $\mathrm{E_{BERT}} = (\mathrm{e}_1, \ldots, \mathrm{e_n})$ for input sequence Z is extracted and the dimension size of each token's feature is 768.

3.3 GPT-3.5 Prediction Module

The GPT-3.5 prediction module aims to leverage the emergent capabilities of GPT-3.5 to identify valid span and entity types. The predictions of GPT-3.5 provide prior knowledge for the local model, which to-some-extent address the data sparsity issue. The process is shown in Figure 3-4. GPT-3.5 relies on pre-defined prompts to generalize and adapt to specific tasks based on the immediate context or instructions within the input prompt. A prompt is essentially an input string to instruct the model to produce desired texts or perform certain tasks. An example of the prompt for NER is shown in Figure 3-4, which consists of four parts: task description, response format, few-shot demonstrations, and input sentence.

Task description clearly outlines the objective of the NER task, assigning a role to GPT-3.5. In response format part, the output format is designated as JSON, a machine-readable format, which can

effectively prevent GPT-3.5 from fabricating spans and entity types. In few-shot demonstrations part, to better utilize the maximum capabilities of GPT-3.5, a few demonstrations are incorporated within the model input. Due to the varying average token counts of demonstrations in different domain datasets, the number of demonstrations in the prompt varies. For the construction dataset, military dataset, and clinic dataset, the respective minimum number of demonstrations for each entity type incorporated in the prompt is 20, 17, and 4. In input sentence part, for each prediction, a data sample from which named entities need to be extracted is placed in the input. Subsequently, GPT-3.5 will return the named entities extracted from the input text according to the response format set in the prompt.

Prompt for NER task

1 Task Description

You are a system capable of accurately recognizing and extracting named entities in the construction domain. You will receive a text, and your task is to identify and extract specified named entities from this text and categorize the extracted entities into predefined entity categories as follows:
Constraints
Attributes
Tasks
Digits

2 Response Format

Your output format must adhere to this template and cannot be in any other format.
Output format:
[{W: Entity extracted from the input text, which must remain unchanged from the original text, L: Entity category selected from the predefined entity categories}, ..., {W: Entity extracted from the input text, which must remain unchanged from the original text, L: Entity category selected from the predefined entity categories}]

3 Few-shot Demonstrations

Example:
Input: After the surface sealing material has solidified, it should appear uniform and smooth, free from any cracks or peeling. 7 days after the injection of the repair adhesive into the cracks, the quality of the grouting should be assessed through core sampling
Output: [{'W': 'sealing material', 'L': 'Constraints'}, {'W': 'injection', 'L': 'Tasks'}, {'W': 'repair adhesive', 'L': 'Constraints'}, {'W': '7 days', 'L': 'Digits'}, {'W': 'grouting', 'L': 'Tasks'}, {'W': 'assessed', 'L': 'Tasks'}]
(The following 10 examples are omitted.)

4 Input Sentence

Please identify and extract the entities from the following input text:
Input: Pressure air test: After the epoxy grout has cured and sealed, a pressure air test is conducted to inspect the seal of the closed area.
Output:

Figure 3-4 An example of the prompt for NER tasks on the construction dataset.

Among GPT APIs of OpenAI, the most powerful one is the gpt-3.5-turbo (i.e., GPT-3.5). To obtain NER results from GPT, the API of GPT-3.5 is applied to send prompts and retrieve model textual responses. The temperature parameter is the key parameter when using GPT-3.5, which controls the randomness of the model's outputs. A higher value makes the outputs more creative and diverse but also more uncertain,

while a lower value makes the outputs more focused and deterministic, sticking closely to the most likely response. The temperature parameter in the API is set to zero, which requires consistent responses from GPT-3.5.

Upon receiving the responses, a customized data parser is developed to interpret responses in JSON format and extract entity types and spans predicted by GPT-3.5. According to the GPT-3.5 predictions, an input sequence is labeled using the BIO tags, similar to the procedure for training the local model. Due to API constraints, the numerical output of GPT-3.5 is inaccessible. Therefore, to fuse the textual output from GPT-3.5 with the BERT embedding in the vector space, each token from the textual outputs of GPT-3.5 is further converted into one-hot vector. In a one-hot encoding scheme, each token is represented as a vector with all zero values except for one element, which corresponds to the entity type belonging to the token, set to one. As such, for the input sequence Z, the prediction of GPT-3.5 O_{GPT} is obtained. The key steps in the GPT-3.5 prediction module are shown in Figure 3-5.

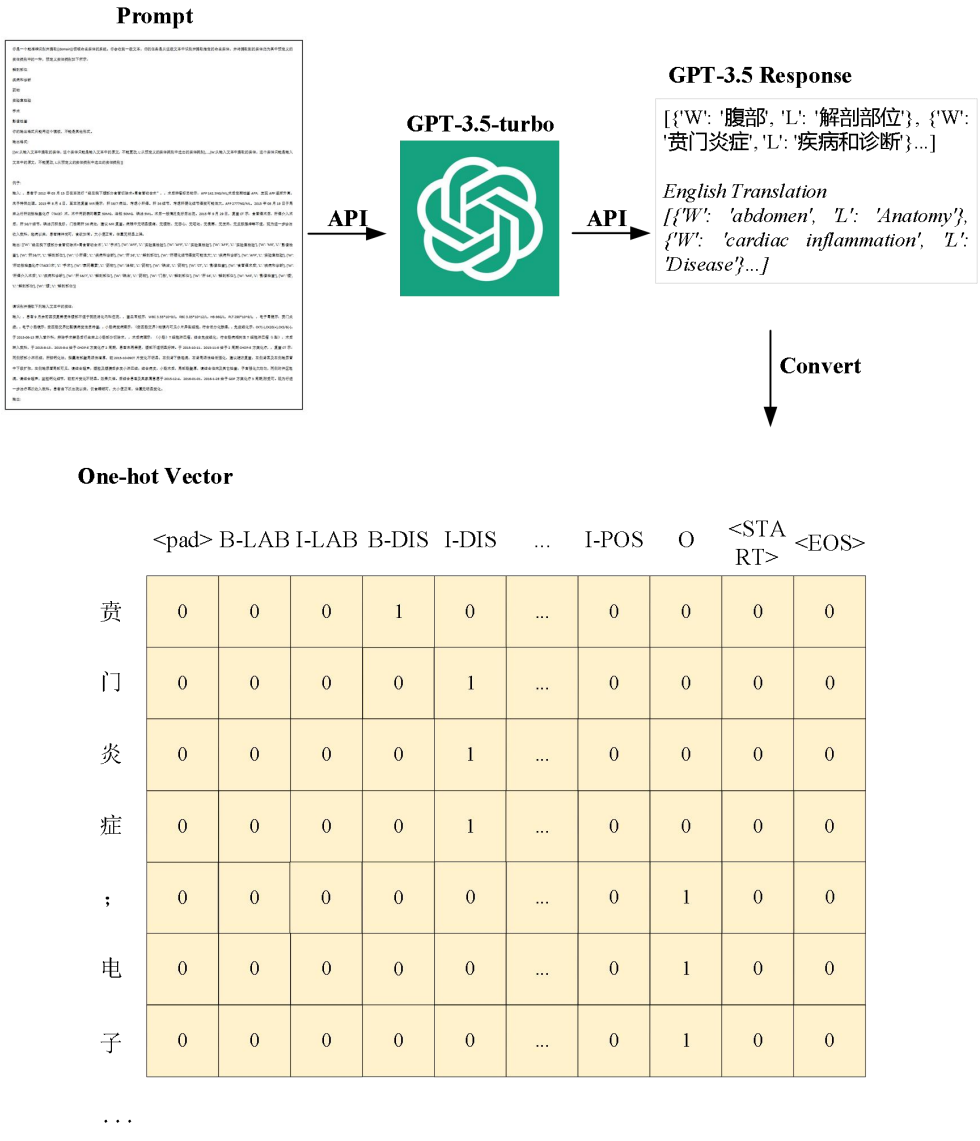

Figure 3-5 Key processes in the GPT-3.5 prediction module. The texts "贲门炎症；电子....." means "cardiac inflammation: electronic.....".

3.4 Fused classification module

A fusion strategy is developed to integrate the prediction outputs from BERT and GPT-3.5. Then, a CFR model is in place to make final NER predictions. Accordingly, the module consists of two layers, fusion layer and CRF layers. Again, as we adopt the featured-based approach, the fused classification module is the only part that requires re-training using domain-specific data.

3.4.1 Fusion Layer

The dynamic embeddings from the frozen pretrained BERT E_{BERT} is transformed into weighted BERT prediction output βO_{BERT} through two fully-connected feed-forward layers. Then, the prediction outputs of GPT-3.5 and BERT are fused by weighted summation (Eq. 7). The weight of O_{GPT}, α, is set to be one, because introducing additional layers with randomly initialized parameters to train α will weaken the

benefits of GPT-3.5 predictions when training data is limited, as the original information is lost when GPT-3.5 predictions are transformed by a randomly initialized matrix. Meanwhile, βO_{BERT} is the weighted BERT outputs obtained from the previous fully-connected feed-forward layers (Eq. 6), where N is the number of BIO tags, d_f is the dimension of dynamic embeddings, and d_h is the dimension of hidden layer. During training, O_{GPT} exerts an influence on the weight β. As α are fixed at one, β trained from the fully connected layers plays a pivotal role in determining the effectiveness of fusing. The process is shown in Figure 3-6.

$$\beta O_{BERT} = \text{Activation}_2(\text{Activation}_1(E_{BERT}W_1 + b_1)W_2 + b_2) \quad (6)$$

$$W1 \in R^{d_f \times d_h}, \quad W1 \in R^{d_h \times N}, \; d_f = 768, d_h = 128$$

$$\text{Fusion} = \alpha O_{GPT} + \beta O_{BERT} \; (\alpha = 1) \quad (7)$$

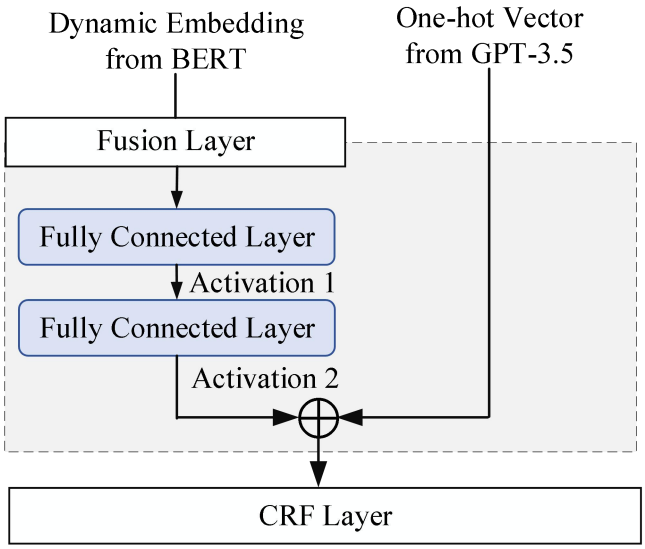

Figure 3-6 Key steps in the fusion strategy

In addition, we discover that the selection of the activation function (Eq. 6) significantly influences the fusing and final NER performance. The element values in O_{GPT} is limited to either 0 or 1, as such, it can only exert impact on βO_{BERT} in a binary manner. After fusion, O_{GPT} can bring both positive effects and negative effects. The negative effect refers to situations where the prediction results of βO_{BERT} are correct, but the predictions of O_{GPT} are wrong. Otherwise, the opposite situation applies. The effects can be better illustrated in the following examples.

In the example of negative effect, the entity type of "贲" predicted by βO_{BERT} is B-DIS and is correct. As such, the probability of B-DIS (*v4*) becomes the largest one among all elements in the output vector of βO_{BERT}. However, O_{GPT} wrongly predicts its entity type as I-DIS; consequently, the probability of I-DIS is

1, while the values of other elements in the vector of O_{GPT} are 0. To obtain the correct entity type, we expect that after fusion, the probability of B-DIS is larger than the probability of I-DIS, i.e., *v4* > *v5*+1. Thus, two inequations in Eq. 8 should be satisfied, and unless the difference between the βO_{BERT} value of ground truth label and that of any other labels is larger than 1, the correct prediction from βO_{BERT} cannot be affected by the wrong prediction from O_{GPT}.

$$\begin{cases} v4 \geq v5 \\ v4 > v5 + 1 \end{cases} \quad (8)$$

$$\text{Thus, } v4 - v5 > 1 \quad (9)$$

		<pad>	B-LAB	I-LAB	B-DIS	I-DIS	...	<EOS>
贲	vector in βO_{BERT}	v_1	v_2	v_3	**v_4**	v_5	...	v_n
	vector in βO_{GPT}	0	0	0	**1**	0	...	0

Figure 3-7 Example of negative effect, where the predicted entity type from weighted BERT prediction is correct, but that from O_{GPT} is wrong

On the other hand, the following case demonstrates the positive effect. The wrong prediction from βO_{BERT} can be amended by the correct prediction from O_{GPT}, only when the difference between the βO_{BERT} value of the false-positive label and the ground truth label is smaller than 1.

$$\begin{cases} v4 \leq v5 \\ v4 + 1 > v5 \end{cases} \quad (10)$$

$$\text{Thus, } 0 \leq v5 - v4 < 1 \quad (11)$$

		<pad>	B-LAB	I-LAB	B-DIS	I-DIS	...	<EOS>
贲	vector in βO_{BERT}	v_1	v_2	v_3	v_4	**v_5**	...	v_n
	vector in βO_{GPT}	0	0	0	**1**	0	...	0

Figure 3-8 Example of positive effect, where the predicted entity type from βO_{BERT} is wrong, but that from O_{GPT} is correct.

The value range of βO_{BERT} is [Min(βO_{BERT}), Max(βO_{BERT})]. Considering the extreme situation in the above two scenarios, the value of v4, v5 is either Min(βO_{BERT}) or Max(βO_{BERT}). To satisfy the inequality Eq. 9 and Eq. 11, the interval value range of βO_{BERT} should follow the criteria below.

$$\text{Max}(\beta\text{OBERT}) - \text{Min}(\beta\text{OBERT}) > 1 \quad (12)$$

According to the equation for βO_{BERT} (Eq. 6), such a criterion leads to the choice of the second activation function is constraint. For instance, Sigmoid function is inappropriate as the second activation function due to its interval range is equal to 1. As such, the search space of the activation function is set to

Tanh, ReLU, Leaky ReLU, and none of activation function (as detailed in section5.1).

3.4.2 CRF layer

The CRF has been widely applied to realize NER, which can be regarded as an undirected graphical model taking immediate neighborhood of a text token into account and predicting the probability of the adjacent field. In particular, the CRF layer calculates scores for different combinations of predicted BIO tags. The core algorithm is shown in Eq. 13, where T is a trainable transition matrix indicating the transfer probability between tags among adjacent tokens, X is the token sequence, and y is the tag sequence.

$$\mathrm{Score}(X, y) = \sum_{i=0}^{n} T_{y_i, y_{i+1}} + \sum_{i=0}^{n} I \qquad (13)$$

The loss function of the CRF layer is illustrated in Eq. 14. The greater the proportion of the score of correct tag sequence to the sum of all scores, the better. According to the loss function, the matrix T is trained through backward propagation. The final tag sequence is the one with the highest score among all possible sequences.

$$-\log(p(y_{gold}|X)) = -\mathrm{score}(X, y) + \log\left(\sum_{y \in Y_x} e^{\mathrm{score}(X,y)}\right) \qquad (14)$$

4. Industry dataset development

To evaluate the performance of GPTBERT, specific datasets of three industries were constructed: clinic, military, and construction, all facing data scarcity and practical challenges in human annotation.

4.1 Construction Dataset

The Construction dataset, developed by Wu et al. [54], is employed for the Chinese construction specifications NER task, focusing on solving constraint management in construction projects. The dataset comprises 949 sentences with four entity types, namely, constraints, attributes, tasks, and digits. The dataset was partitioned into three splits: 770 sentences for training, 85 sentences for validation, 94 sentences for testing. With the assistance of GPT-3.5, a total of 568 inaccurately annotated spans were corrected (as illustrated in Section 5.4.1). The distribution of entity types is shown in the Table 4-1. Through tokenization, sentences were further broken-down to Chinese character-level tokens. Accordingly, labels were transformed into the BIO format given as input to our model.

Table 4-1 Entity distribution of construction dataset

	Constraints	Attributes	Tasks	Digits
Training Set	2,451	527	1,593	360
Validation Set	223	62	156	43
Test Set	308	84	179	67
Total	2,982	673	1,928	470

4.2 CCKS2019-Clinic Dataset

To fill the gap in datasets of Chinese electronic clinic records, the China Conference on Knowledge Graph and Semantic Computing (CCKS) 2019 developed the CCKS2019-Clinic dataset. This dataset comprised 1000 samples in paragraph granularity, which were extracted from actual electronic clinic records. There were six entity types: diagnosis, image examination, laboratory examination, operation, medicine, and anatomic site. The original samples had varying token lengths; some exceeded the max token length limitation in BERT (i.e., 512). Thus, the samples were further split into sentences using the full stop. As such, 6975 sentences were generated for training, 775 for validation, and 2480 for testing. Table 4-2 presents the distribution of different entity types. Tokenization and BIO format transformation were then conducted on the CCKS2019-Clinic Dataset.

Table 4-2 Entity distribution of CCKS2019-Clinic dataset

	Disease	Image Examination	Laboratory examination	Operation	Medicine	Anatomy
Training Set	3,569	838	1,206	793	1,587	8,927
Validation Set	311	26	36	90	231	947
Test Set	1,185	252	607	128	483	3,723
Total	5,065	1,116	1,849	1,011	2,301	13,597

4.3 CCKS2020-Military Dataset

To promote the technology of NER in the field of military test and evaluation, CCKS and the System Engineering Research Institute of the Academy of Military Sciences (AMS) released the NER dataset and launched an online challenge in 2020, aiming to recognize four entity types: test elements, performance indicators, system components, and mission scenarios. Among the three datasets, the CCKS2020-Military dataset was the smallest in size due to confidentiality issues. Only 400 labeled data were published. The dataset also involved a large number of military terminologies, which were uncommon in the general domain corpus. As a result, the SOTA F1 score on this dataset was around 72%, which was achieved by self-constructing a military corpus crawled from websites to pre-train a domain-specific language model. As our experiment focused on testing the robustness of GPTBERT, instead of additional pretraining on domain datasets, we adopted the pre-trained BERT on the general domain corpus. The model's performance on this dataset can be considered a stringent evaluation of the model's generalization ability in a small data domain. As the CCKS did not publish the testing data, the training data was partitioned into three groups: 40 samples for testing, 36 samples for validation, and 324 samples for training, where each data sample consists of one or more sentences. The distribution of entity types is illustrated in Table 4-3. Tokenization and BIO format transformation were then conducted on the CCKS2020-Military Dataset.

Table 4-3 Entity distribution of CCKS2020-Military dataset

	Test Elements	Performance Indicators	System Components	Mission Scenarios
Training Set	1,486	643	359	506
Validation Set	160	64	44	47
Test Set	181	75	40	47
Total	1,827	782	443	600

5. Experiments

In this section, the implementation details and evaluation metrics are first introduced. Then, the effectiveness of GPTBERT is verified with an ablation study.

5.1 Implementation details

The GPTBERT was trained by a single NVIDIA GeForce RTX 4090 GPU. Table 4-4 shows the optimal hyper-parameters for the model on each dataset. The hyper-parameters were tuned based on grid searching. According to the analysis in Section 3.4.1, the search space of the activation function is among Tanh, ReLU, Leaky ReLU, and none of the activation functions. Through evaluating the performance on the validation dataset, the hyperparameters with the highest F1-score were selected.

Table 4-4 Results of hyper-parameters tuning.

	Construction	Military	Clinic
Max Length	128	512	512
Batch Size	4	4	16
Learning Rate	0.001	0.001	0.001
Weight Decay	0.0005	0.0005	0.0005
Patience	5	5	5
Hidden Size	128	256	128
Activation 1	Tanh	Tanh	Tanh
Activation 2	None	None	None

5.2 Evaluation Metrics

In some datasets, when the same entity word appeared multiple times in the given input text, it was annotated only once, while in other datasets, it was annotated multiple times. The annotation standard was unified so that the span was annotated as many times as it appeared in the text. The prediction result was regarded as correct only when the span and entity type of prediction exactly matched the ground truth. The precision rate (P), recall rate (R), and F1 score (F1) were adopted as evaluation metrics, where TP, FP, and FN represent true positives, false positives, and false negatives, respectively.

$$\text{Precision Rate (P)} = \frac{\text{true postives}}{\text{true postives} + \text{false positives}} \quad (15)$$

$$\text{Recall Rate (R)} = \frac{\text{true postives}}{\text{true positives} + \text{false negatives}} \quad (16)$$

$$\text{F1 Score (F1)} = \frac{2\times P\times R}{P+R} \quad (17)$$

5.3 Results

To verify the robustness of GPTBERT, in addition to testing its performance, two comparative experiments were designed. These experiments substituted the pre-trained model, BERT, in the Dynamic embedding module of GPTBERT with two derivative models of BERT: NEZHA and RoBERTa. These two new methods are respectively referred to as GPTNEZHA and GPTRoBERTa. Collectively, these two methods and GPTBERT are referred to as "Our approach." NEZHA and RoBERTa are variants of BERT and each makes certain improvements. NEZHA introduces functional relative positional encoding, which allows for capturing longer-range dependencies and is specifically optimized for Chinese language understanding. RoBERTa extends BERT by training for a longer period on a larger dataset with dynamic masking and by removing the Next Sentence Prediction task, thereby enhancing the overall performance on language understanding tasks. Additionally, to validate the effectiveness of our approach, two sets of ablation experiments were designed: the feature-based approach (i.e., the conventional approach without leveraging predictions from GPT-3.5) and the fine-tuning approach. Each set of ablation experiments utilized three pre-trained models: BERT, NEZHA, and RoBERTa. To demonstrate that our approach excelled over the feature-based approach in low-data scenarios, the performance of our approach and the feature-based approach were tested across varying training data sizes. Specifically, the training started with zero samples and incrementally increased the training sample size until the full training dataset was utilized. For the military and construction datasets, the training data size was increased by 20 each time. While for the relatively larger clinic dataset, the training data size was increased by 300 each time. To better observe the model's performance on small training datasets, when the training data size of the clinic dataset was less than 300, the size was increased by 50 each time. Tables 5-2 and Figures 5-1 demonstrate the respective performance metrics. Both the feature-based approach and our approach kept the PLM's parameters frozen during the training phase, resulting in an equal number of trainable parameters for the two methods. On the other hand, the fine-tuning approach employed a full parameter fine-tuning strategy, which updated all the PLM's parameters during the training stage. This led to a significantly larger number of trainable parameters than other methods in the experiments. Table 5-1 elucidates the number of trainable parameters for each method.

Table 5-1 Number of trainable parameters of different models on different datasets

Dataset	Our Approach	Proportion of total parameters	Feature-based Approach	Fine-tuning Approach

Construction	100,124[①②③]	0.10%[①②③]	100,124[①②③]	102,305,222[①] 103,287,492[②③]
Military	200,092[①②③]	0.19%[①②③]	200,092[①②③]	103,586,886[①] 104,569,156[②③]
Clinic	100,752[①②③]	0.10%[①②③]	100,752[①②③]	102,305,970[①] 103,287,604[②③]

Note: ①,②,③ refer to the PLMs using NEZHA, RoBERTa, and BERT, respectively.

Table 5-2 Model Performance metrics on different datasets and different sizes of training set

Dataset	Model	F1 Score (training data size = 0)	F1 Score (training data size = 40)	F1 Score (training data size = 760)	F1 Score (all training data)
Construction	Our Approach	50.13%[①]	57.99%[①]	84.67%[①]	83.23%[①]
		48.62%[②]	59.77%[②]	83.58%[②]	**85.51%**[②]
		31.48%[③]	**62.70%**[③]	**86.20%**[③]	84.96%[③]
	Feature-based Approach	0.31%[①]	54.83%[①]	82.27%[①]	82.35%[①]
		0.09%[②]	53.55%[②]	81.33%[②]	83.68%[②]
		0.13%[③]	55.54%[③]	83.15%[③]	83.58%[③]
	Fine-tuning Approach	-	-	-	84.23%[①]
					83.76%[②]
					85.25%[③]
	GPT-3.5	**56.51%**	-	-	-
Dataset	Model	F1 Score (training data size = 0)	F1 Score (training data size = 40)	F1 Score (training data size = 300)	F1 Score (all training data)
Military	Our Approach	36.82%[①]	30.76%[①]	47.15%[①]	51.66%[①]
		36.58%[②]	31.48%[②]	53.41%[②]	53.38%[②]
		34.88%[③]	**34.58%**[③]	**57.94%**[③]	**57.20%**[③]
	Feature-based Approach	0.19%[①]	23.52%[①]	48.54%[①]	49.28%[①]
		0.00%[②]	17.47%[②]	53.44%[②]	52.58%[②]
		0.00%[③]	11.46%[③]	53.11%[③]	50.20%[③]
	Fine-tuning Approach	-	-	-	54.36%[①]
					51.03%[②]
					51.32%[③]
	GPT-3.5	**47.68%**	-	-	-
Dataset	Model	F1 Score (training data size = 0)	F1 Score (training data size = 50)	F1 Score (training data size = 100)	F1 Score (all training data)
Clinic	Our Approach	**32.18%**[①]	0.78%[①]	**52.05%**[①]	78.00%[①]
		32.07%[②]	**42.03%**[②]	48.96%[②]	76.81%[②]
		27.94%[③]	41.52%[③]	48.89%[③]	76.91%[③]
	Feature-based Approach	0.02%[①]	0.00%[①]	47.49%[①]	76.04%[①]
		0.00%[②]	0.00%[②]	44.47%[②]	74.89%[②]
		0.00%[③]	0.00%[③]	47.07%[③]	75.71%[③]
	Fine-tuning	-	-	-	77.68%[①]

Approach				74.75%②
				78.07%③
GPT-3.5	29.78%	-	-	-

Note: ①,②,③ refer to the PLMs using NEZHA, RoBERTa, and BERT, respectively. The bolded text indicates the model that performed best under a certain training data size.

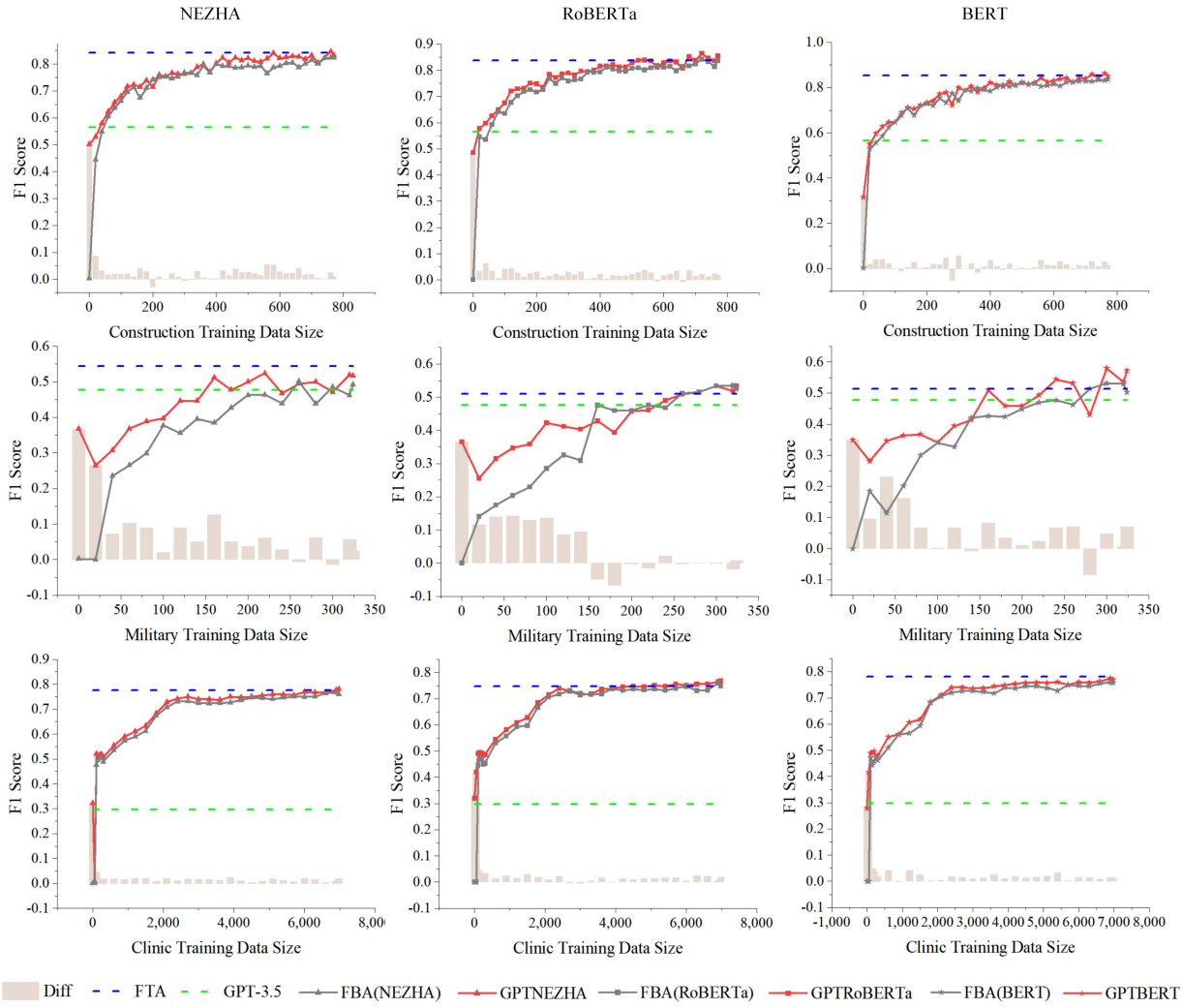

Figure 5-1. The performance of different models across datasets from various industries and at different scales of training data. In these charts, each column represents a different PLM, ordered as NEZHA, RoBERTa, and BERT. Each row corresponds to the results for a different industry dataset, specifically for the construction, military, and clinic sectors in sequence. The gray line and red line represent the performance (measured by F1 score) of the feature-based approach and our approach on the test set after training on different sizes of training sets. Bars indicate the difference in performance between our approach and the feature-based approach on the test set. The Blue dashed line represents the performance (F1 score) of fine-tuning approach training on the complete training set. The Green dashed line represents the performance (F1 score) of the plug-and-play GPT-3.5 model on the test set.

When the training dataset was extremely limited (<=50 samples), the classical methods easily suffered from biases because the small data could not capture the full range of variability in the problem space; however, our approach leveraged the zero-shot learning capabilities of GPT-3.5, thereby still demonstrating competency in the NER task and significantly outperforming the feature-based approach. Moreover, our approach achieved superior performance over GPT-3.5 on the construction and clinic datasets with a comparatively small number of training samples. For example, on the construction dataset, our approach exceeded GPT-3.5 with just 40 training samples. On the clinic dataset, both GPTRoBERTa and GPTNEZHA outperformed GPT-3.5 without the need for any labeled data (i.e., the zero-shot learning scenario). GPTBERT, with a mere 50 training samples, also surpassed GPT-3.5, whereas the feature-based approach needed twice of training data to reach similar results.

Table 5-3 Boosting effect of different models on different datasets

Dataset	Model	Max	Min	Average	Median
Construction	GPTNEZHA	49.82%	-2.89%	3.20%	2.00%
	GPTRoBERTa	48.52%	-0.74%	3.30%	2.17%
	GPTBERT	31.34%	-5.11%	2.16%	1.46%
Military	GPTNEZHA	36.64%	-1.39%	8.24%	5.88%
	GPTRoBERTa	36.58%	-6.64%	6.07%	1.47%
	GPTBERT	34.89%	-8.44%	7.19%	6.67%
Clinic	GPTNEZHA	32.16%	0.38%	2.54%	1.61%
	GPTRoBERTa	42.04%	-0.52%	4.03%	1.74%
	GPTBERT	41.53%	-0.08%	3.98%	1.53%

Our approach's superiority over the feature-based approach was almost universally observed across three industrial datasets. While GPT-3.5 showed underwhelming results on the clinic dataset, our approach almost consistently outperformed the feature-based approach regardless of the training data sizes. We define the difference in F1 scores between our approach and the feature-based approach as the "boosting effect". This effect is represented by the "diff" values in Figures 5-1 and further detailed in Table 5-3. According to the statistics presented in Table 5-3, the average and median values of the boosting effect are generally above zero across the datasets. This indicates that 1) GPT-3.5 and the feature-based approach learn from different sample distributions; 2) the structure of our approach is beneficial as it integrates predictions from GPT-3.5 with those from the feature-based approach. However, as the number of training samples increases, the boosting effect tends to converge within a certain range. In the military dataset, the boosting effect fluctuated, converging at less than 7%; for the clinic and construction datasets, given the relatively sufficient number of training samples, the boosting effect tended to reduce to around 4%.

When trained on the entire dataset, the performance of our approach reached or exceeded that of the fine-tuning approach which updating all model parameters. However, the performance of the feature-based approach generally fell short compared to fine-tuning, regardless of the increasing size of the training data. Our method achieved the best results across military and construction datasets and was almost identical to the best results in the clinic dataset. In the construction dataset, the highest F1 score achieved was 86.21%, by GPTBERT with 760 training samples. In the military dataset, the highest F1 score was 57.95%, by GPTBERT with 300 training samples. For the clinic dataset, the top F1 score was 78.07%, while the second F1 score was 78% achieved by GPTNEZHA with 6,975 training samples.

In summary, our approach consistently delivered better F1 scores than the feature-based approach across various dataset sizes in three different industries. Particularly on training datasets with fewer than 50 samples, our approach demonstrated superior performance over the feature-based approach. On each industrial dataset, our approach achieved the best results, surpassing GPT-3.5, the feature-based and fine-tuning approaches. This indicates the GPTBERT framework is adaptable to BERT variants like NEZHA and RoBERTa, offering a consistent boosting effect. The effectiveness of our fusion strategy, which integrates the LLM GPT-3.5 with localized PLM, is thus confirmed.

5.4 Performance Explanation

The effectiveness of GPTBERT has been validated, and the boosting effect by using GPT-3.5 is demonstrated. In this section, the effect is further analyzed from two aspects. First, a detailed investigation of GPT-3.5's performance across the three datasets was conducted to understand its limitations and potential contributions in NER when combined with the feature-based approach. Second, the predictions on the testing dataset among GPTBERT, the feature-based approach, and GPT-3.5 were compared to analyze both the positive and negative effects resulting from fusing GPT-3.5 predictions.

5.4.1 Performance analysis of GPT-3.5

GPT-3.5 and GPT-4 are the SOTA LLMs. They exhibit remarkable emergent capabilities, excelling in small-sample or zero-shot learning. Consequently, GPT-3.5 is expected to handle NER tasks in specific domains with only a few annotated data while not requiring additional tuning. However, our experiments showed that GPT-3.5 did not perform well across the three domain-specific Chinese datasets. Particularly, the precision rates of GPT-3.5 were less than 50% on the datasets. However, in our experimental settings, the prediction was considered to be correct only when the ground truth and the model prediction had an exact boundary of span and entity type. We supposed that the criterion could be too strict to evaluate the

NER performance of GPT-3.5. On the other hand, partially correct results could also provide useful information in industrial NER applications. For instance, human engineers can easily correct the entity if the predicted boundary is shorter or longer than the true label. Similarly, Han et al. [18]observed that employing a span hard-matching strategy to evaluate ChatGPT was considered inappropriate due to its human-like response generation. As a result, they introduced a soft-matching evaluation strategy to provide a more precise assessment of ChatGPT's performance. In this study, six scenarios that can occur in real-world implementation were listed to further analyze GPT-3.5 performance and errors caused by pervasive issues in LLMs, such as hallucination, inconsistency, and overstatements [55]. Again, "span" was used to denote the target information to be extracted and "types" to indicate entity types. The statistical result of each scenario is illustrated in Table 5-4.

Scenario I. Fabricated Span

The predicted span was a fabrication that was not present in the input text.

GPT-3.5 prediction: {Word：裂缝灌注, Entity Type: 工序任务}

English Translation: {Word： crack injection, Entity Type: Tasks}

Golden Standard: {Word：裂缝灌浆, Entity Type: 工序任务}

English Translation: {Word： crack grouting, Entity Type: Tasks}

Scenario II. Fabricated Type

The predicted entity type was a fabrication that did not exist in predefined types. For instance, the label type "Demolition Tool" returned by GPT-3.5 does not exist among the dataset's predefined label types: Constraints, Attributes, Tasks, and Digits.

GPT-3.5 prediction: {Word：切割机, Entity Type: 拆除机具}

English Translation: {Word： cutting machine, Entity Type: demolition tool}

Scenario III. Partially Correct Boundary with Correct Type

The predicted entity type was accurate, but the predicted boundary was partially correct.

Input Text	裂	缝	灌	浆	完	成
	Crack grouting is completed					
GPT-3.5 Prediction	O	O	I-TP	I-TP	O	O
Golden Standard	B-TP	I-TP	I-TP	I-TP	O	O

Scenario IV. Correct Boundary with Wrong Type

The predicted span was correct, but the predicted type was wrong.

Input Text	裂	缝	灌	浆	完	成
	Crack grouting is completed					
GPT-3.5 Prediction	B-CONS	I-CONS	I-CONS	I-CONS	O	O
Golden Standard	B-TP	I-TP	I-TP	I-TP	O	O

Scenario V. Other Errors

Other types of error were not included in the above scenarios.

Scenario VI. Correct

Both span and entity type were correct.

Table 5-4 A statistical result shows the frequency of GPT-3.5 predicted result in various scenarios. "Num" means the number of spans predicted by GPT-3.5, while "Ratio(%)" means the corresponding percentage. The performance of GPT-3.5 is evaluated on the entire dataset, not just the test set.

Scenario Type	Construction		Military		Clinic	
	Num	Ratio(%)	Num	Ratio(%)	Num	Ratio(%)
Fabricated Span	4	0.0	43	0.9	0	0.0
Fabricated Type	8	0.1	10	0.2	0	0.0
Partially correct boundary with correct entity type	959	11.0	522	11.4	4,802	11.9
Correct boundary with wrong entity type	234	2.7	712	15.6	83	0.2
Other errors	3,468	39.9	1,665	36.4	24,436	60.6
Correct	4,010	46.2	1,624	35.5	10,971	27.2
Total	8,683	100.0	4,576	100.0	40,292	100.0

As GPT-3.5 is a generative model but not a classification model, there is a suspicion that it might fabricate a span or entity type. After multiple trials, we discovered that by configuring the response format in the prompt to adhere to a machine-readable structure, such as JSON, the hallucination of LLMs was mitigated. The statistical results for Scenario I and Scenario II in Table 5-4 presents the outcomes after adjusting the output format to JSON. Scenario III was a prevalent error type in each dataset when using LLMs. The data samples were tested using GPT-4 via web UI for predictions, while the Scenario III error became even more prominent due to its inclination to produce more elaborate responses. In other words, the predicted spans often extend beyond the ground truth. However, as mentioned, although the predicted spans were only partially accurate, they could still offer valuable insights for the local feature-based small model. During dataset construction, errors in Scenario IV would occur if the entity spans were wrongly labeled by humans. Human annotation errors could be attribute to carelessness or cognitive biases, as different individuals had varying understandings of a certain entity definition. When GPT-3.5 was initially

employed to conduct NER on the construction dataset, GPT-3.5 aided in identifying 568 inaccurately annotated spans, accounting for 6.5% of the dataset. For the construction dataset, the value of scenario IV shown in Table 5-4 reflected the performance after correction. As military and clinic datasets were public competition datasets, to ensure the results from GPTBERT were objective, no corrections were applied. This phenomenon indicates that GPT-3.5 could be used for examining manually annotations. In summary, the performance of GPT-3.5 is not optimal, but it is undeniable that GPT-3.5 possesses universal knowledge, allowing it to adapt to various industries without further training. Thus, through a fusing layer, GPTBERT can maximize the utilization of predictions from GPT-3.5 belonging to Scenario III and Scenario VI, thereby reaping the strengths of the SOTA AI model.

5.4.2 Affecting mechanism of the boosting effect

The predictions of all tokens, including those labeled as "Other", were collected from GPTBERT, the feature-based approach, and GPT-3.5 on the testing dataset. For each approach, the predictions were categorized into "correct" and "wrong". The comparative analysis is depicted in Table 5-5 and Figure 5-2. Through the fusing layer, GPT-3.5 had both positive and negative impacts on the performance of GPTBERT. The positive impact of GPTBERT stemmed from tokens where accurate predictions of GPT-3.5 corrected the errors of the feature-based approach and vice versa. Overall, after fusion, the positive impact of GPT-3.5 prevailed over the negative.

For instance, in the military dataset, with 324 training samples, the F1 Score of GPTBERT was 7% higher than that of the feature-based approach. GPTBERT rectified 210 tokens that the feature-based approach had wrongly predicted, where 188 of the corrections were attributed to the precise predictions of GPT-3.5. In contrast, GPTBERT mistakenly modified 131 tokens that the feature-based method had predicted correctly, all of which were swayed by inaccuracies of GPT-3.5. The findings indicate that the positive and negative impacts of GPT-3.5 on GPTBERT are essential drivers of the boosting effect of GPTBERT. By carefully selecting the activation functions in the fusing layer, GPTBERT could minimize the adoption of incorrect predictions from GPT-3.5 while maximizing the usage of correct predictions. In the aforementioned example, 748 tokens were predicted incorrectly by GPT-3.5, which were predicted correctly by the feature-based method. After being fused in GPTBERT, only 131 of the 748 GPT-3.5 predictions had negative impacts (17.51%), while the negative impacts of another 617 GPT-3.5 predictions were avoided (82.48%). In contrast, there were 284 tokens that GPT-3.5 predicted correctly, but the feature-based method predicted incorrectly. After the fusion, 188 of these 284 GPT-3.5 predictions had

positive impacts (66.19%), and another 96 GPT-3.5 predictions belonged to ineffective correction (33.80%).

Table 5-5 Comparison of the token prediction performance among three models on the military test set.

	Feature-based Approach Correct		Feature-based Approach Wrong	
	Num of Tokens	Ratio (%)	Num of Tokens	Ratio (%)
GPT-3.5 Correct	3,108	69.04	284	6.31
GPT-3.5 Wrong	748	16.61	362	8.04
GPTBERT Correct	3,725	82.74	210	4.66
GPTBERT Wrong	131	2.91	436	9.68

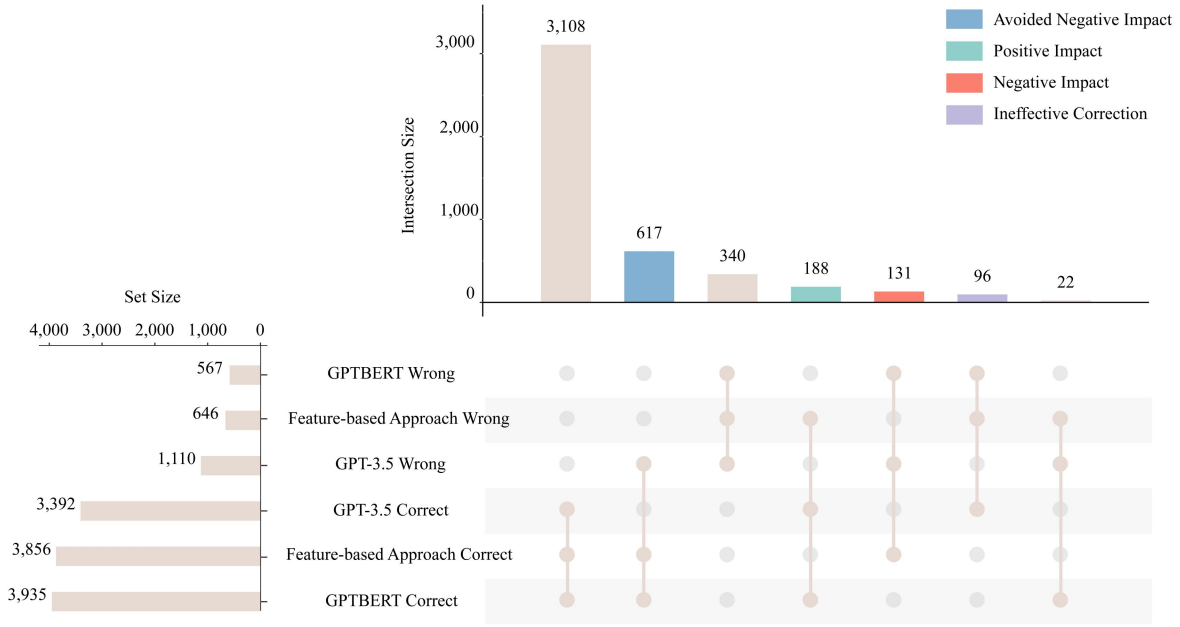

Figure 5-2. Comparison of the token prediction performance among three models in the military test set. Both GPTBERT and the feature-based approach were trained on 324 training samples. The figure illustrates the size of the intersections among six sets, including the set of tokens correctly predicted by GPT-3.5, the set of tokens incorrectly predicted by GPT-3.5, the set of tokens correctly predicted by GPTBERT, the set of tokens incorrectly predicted by GPTBERT, the set of tokens correctly predicted by the feature-based approach, the set of tokens incorrectly predicted by the feature-based approach.

6. Discussion

This section discusses the primary contributions of GPTBERT. First, GPTBERT effectively reduces computational resources while still achieving good results. GPTBERT presents an innovative fine-tuning method, freezing the remote LLM and fine-tuning a small portion of parameters in the local BERT. In this way, GPTBERT outperforms both the local model and LLM on domain-specific datasets. GPTBERT only requires updating less than 0.2% of parameters (see Table 5-1) in BERT, which can match or exceed the

performance of full fine-tuning. SOTA NER models, especially PLMs and LLMs, can be computationally demanding, but in practical industries, only CPUs or low-end GPUs are accessible. By reducing the computational needs without compromising performance, GPTBERT significantly lowers the barriers to NER implementation in the industry. In addition, the conventional feature-based approach similarly updates only about 0.2% of full parameters, but the performance consistently underperforms compared to the fine-tuning approach and is generally inferior to GPTBERT. In essence, GPTBERT reaps the benefits from GPT-3.5 and the feature-based approach that learns data features from varying sample distributions. It harnesses the benefits by incorporating the predictions from GPT-3.5 on top of the feature-based method, enhancing the performance of the feature-based approach based on GPT-3.5 predictions. Thus, GPTBERT strikes a balance between applying LLMs as a versatile tool to solve NER in all industries and developing specialized local models through full-parameter fine-tuning. Incorporating GPT-3.5 introduces both positive and negative impacts. However, by meticulously choosing activation functions in the fusion layer, GPTBERT successfully minimizes incorrect predictions and capitalizes on the correct ones, ensuring an overall performance boost after integration.

Second, GPTBERT exploits the generalization capabilities in GPT-3.5, lessening reliance on training data. Although the performance of GPT-3.5 is not always optimal, its expansive knowledge base ensures adaptability across varied industries without further updating parameters. GPTBERT particularly shines against the feature-based approach in low-data scenarios, especially where the training dataset is extremely limited. In practice, many industries lack the annotated datasets that are vital for domain-specific NER model training. Creating and annotating the domain datasets are labor-intensive and require specific annotation guidelines developed by domain experts. Inconsistencies in the interpretation of annotation rules can lead to annotation errors. Some domain data, like clinic data, is private, making dataset establishment more challenging. Additionally, GPTBERT also works without creating sophisticated prompts or instructions, which is often required in several recent studies that fine-tune LLM using prompts, e.g., FLAN-T5 [19]. By minimizing its reliance on training datasets and prompts, GPTBERT decreases deployment costs and improves its adaptability in various industries.

Third, GPTBERT overcomes challenges tied to the API of GPT-3.5. Specifically, GPT-3.5 is a closed-source model. Hence, existing studies adopting GPT-3.5 in downstream applications can only explicitly use text responses instead of accessing hidden vectors in its intermediate layers and optimizing its structures. The GPTBERT proposes a solution to take use of the outputs of GPT-3.5 more flexibly.

Specifically, GPTBERT transforms the text response from GPT-3.5 API into a one-hot vector. In this way, GPTBERT assigns numerical features to the unstructured texts, which can then be more effectively fused with the output representation vector from the local BERT model.

Nevertheless, there are several limitations to the proposed approach. For spans in which GPT-3.5 correctly predicted the boundaries but inaccurately predicted the types, GPTBERT did not utilize this valuable boundary information. This paper aims to validate the possibility and effectiveness of fusing text responses from GPT-3.5 with the text embeddings generated by the local BERT. Hence, no further optimization was made for the local model for specified-domain tasks, and no post-processing was conducted for various tasks. Thus, there is room for improvement in the performance of GPTBERT. In addition, future research could delve deeper into its adaptability across diverse NLP tasks and the potential to fuse other pre-trained models to further enhance performance.

7. Conclusion

When implementing NER in practical applications, it often faces challenges due to the excessive requirements of high-quality data and computing resources. To address the issues, we propose the novel NER method named GPTBERT. GPTBERT first extracts text embeddings from input sentences using a pre-trained BERT model. Then, via the Open-AI API and a specific prompt, the named entities in the input sentences are identified by GPT-3.5. The predictions are transformed into one-hot vectors. Subsequently, we fused one-hot vectors from GPT-3.5 and text embeddings generated by the local BERT. Finally, a CRF layer is employed to predict entity types. The essence of GPTBERT is to incorporate prior knowledge from the SOTA generative model, GPT-3.5, into a local model through a fusion layer. This strategy makes GPTBERT feasible and affordable across various industries without sacrificing performance. It can be argued that GPTBERT bridges the strengths of GPT-3.5 and BERT, effectively combining the few-shot learning capability of GPT-3.5 with the low-cost fine-tuning of the feature-based approach that requires updating only a few parameters. Through comprehensive experiments, we verify that the performance of GPTBERT consistently surpasses that of the feature-based approach across training samples of various scales and domains. After fine-tuning with a small amount of data, the GPTBERT can easily exceed that of GPT-3.5; moreover, on certain datasets, the GPTBERT surpasses the full-parameter fine-tuning approach. Therefore, the GPTBERT can enhance the practicality of PLMs and LLMs in solving real-world engineering problems across various industries, thereby facilitating various downstream tasks such as information retrieval, compliance checking, and automatic answering.

Reference

[1] B. Jehangir, S. Radhakrishnan, and R. Agarwal, "A survey on Named Entity Recognition—datasets, tools, and methodologies," *Natural Language Processing Journal,* vol. 3, p. 100017, 2023.

[2] T. Schopf, K. Arabi, and F. Matthes, "Exploring the landscape of natural language processing research," *arXiv preprint arXiv:.10652,* 2023.

[3] M. A. Khalid, V. Jijkoun, and M. De Rijke, "The impact of named entity normalization on information retrieval for question answering," in *European Conference on Information Retrieval*, 2008, pp. 705-710: Springer.

[4] Q. Zhang, C. Xue, X. Su, P. Zhou, X. Wang, and J. Zhang, "Named entity recognition for Chinese construction documents based on conditional random field," *Frontiers of Engineering Management,* vol. 10, no. 2, pp. 237-249, 2023.

[5] A. Kumar and B. Starly, ""FabNER": information extraction from manufacturing process science domain literature using named entity recognition," *Journal of Intelligent Manufacturing,* vol. 33, no. 8, pp. 2393-2407, 2022.

[6] S. Moon, G. Lee, S. Chi, and H. Oh, "Automated construction specification review with named entity recognition using natural language processing," *Journal of Construction Engineering Management,* vol. 147, no. 1, p. 04020147, 2021.

[7] X. Yang, D. Peng, and Z. Hou, "Named Entity Recognition in Military Weapons Domain with Multi-neural Networks Incorporating Relative Position Information," in *2022 3rd International Conference on Pattern Recognition and Machine Learning (PRML)*, 2022, pp. 346-351: IEEE.

[8] M. Zhou, J. Tan, S. Yang, H. Wang, L. Wang, and Z. Xiao, "Ensemble transfer learning on augmented domain resources for oncological named entity recognition in Chinese clinical records," *IEEE Access,* 2023.

[9] A. Akbik, D. Blythe, and R. Vollgraf, "Contextual string embeddings for sequence labeling," in *Proceedings of the 27th international conference on computational linguistics*, 2018, pp. 1638-1649.

[10] Z. Shan, L. Rui, and C. Zhiping, "Survey of Chinese named entity recognition," *Journal of Frontiers of Computer Science Technology,* vol. 16, no. 2, p. 296, 2022.

[11] S. Hochreiter and J. Schmidhuber, "Long Short-term Memory," *Neural computation,* vol. 9, pp. 1735-80, 1997.

[12] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555,* 2014.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez *et al.*, "Attention is all you need," *Advances in neural information processing systems,* vol. 30, 2017.

[14] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee *et al.*, "Deep contextualized word representations," presented at the Conference of the North American Chapter of the Association for

Computational Linguistics, 2018. Available: https://ui.adsabs.harvard.edu/abs/2018arXiv180205365P

[15] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

[16] J. D. M.-W. C. Kenton and L. K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of naacL-HLT*, 2019, vol. 1, p. 2.

[17] J. Li, A. Sun, J. Han, and C. Li, "A survey on deep learning for named entity recognition," *IEEE Transactions on Knowledge Data Engineering,* vol. 34, no. 1, pp. 50-70, 2020.

[18] R. Han, T. Peng, C. Yang, B. Wang, L. Liu, and X. Wan, "Is Information Extraction Solved by ChatGPT? An Analysis of Performance, Evaluation Criteria, Robustness and Errors," *arXiv preprint arXiv:.14450,* 2023.

[19] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus *et al.*, "Scaling instruction-finetuned language models," *arXiv preprint arXiv:.11416,* 2022.

[20] E. Buber and D. Banu, "Performance analysis and CPU vs GPU comparison for deep learning," in *2018 6th International Conference on Control Engineering & Information Technology (CEIT)*, 2018, pp. 1-6: IEEE.

[21] Y. Bengio, R. Ducharme, and P. Vincent, "A neural probabilistic language model," *Advances in neural information processing systems,* vol. 13, 2000.

[22] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems,* vol. 26, 2013.

[23] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the association for computational linguistics,* vol. 5, pp. 135-146, 2017.

[24] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532-1543.

[25] Y. Sun, S. Wang, Y. Li, S. Feng, X. Chen, H. Zhang *et al.*, "Ernie: Enhanced representation through knowledge integration," *arXiv preprint arXiv:.09223,* 2019.

[26] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," *arXiv preprint arXiv:.11942,* 2019.

[27] J. Wei, X. Ren, X. Li, W. Huang, Y. Liao, Y. Wang *et al.*, "Nezha: Neural contextualized representation for chinese language understanding," *arXiv preprint arXiv:.00204,* 2019.

[28] H. Wang, J. Li, H. Wu, E. Hovy, and Y. Sun, "Pre-trained language models and their applications," *Engineering,* 2022.

[29] C. Gong, J. Tang, S. Zhou, Z. Hao, and J. Wang, "Chinese named entity recognition with bert," *DEStech transactions on computer science engineering,* vol. 12, 2019.

[30] X. Mengge, B. Yu, T. Liu, Y. Zhang, E. Meng, and B. Wang, "Porous lattice transformer encoder for Chinese NER," in *Proceedings of the 28th international conference on computational linguistics*, 2020, pp. 3831-3841.

[31] R. Ma, M. Peng, Q. Zhang, and X. Huang, "Simplify the usage of lexicon in Chinese NER," *arXiv preprint arXiv:.05969,* 2019.

[32] Y. Lin, L. Liu, H. Ji, D. Yu, and J. Han, "Reliability-aware dynamic feature composition for name tagging," in *Proceedings of the 57th annual meeting of the association for computational linguistics*, 2019, pp. 165-174.

[33] Y. Lu, R. Yang, D. Zhou, H. Xiang, and C. Yin, "A military named entity recognition method combined

with dictionary," in *Proceedings of the 2019 2nd international conference on algorithms, computing and artificial intelligence*, 2019, pp. 591-596.

[34] I. M. Soriano and J. L. C. Peña, "STMC: Semantic tag medical concept using word2vec representation," in *2018 IEEE 31st International Symposium on Computer-Based Medical Systems (CBMS)*, 2018, pp. 393-398: IEEE.

[35] S. Jian-wei, D. Yi-chuan, and S. Cheng, "Research on named entity recognition of construction safety accident text based on pre-trained language model," *Journal of Graphics,* vol. 42, no. 2, p. 307, 2021.

[36] Z. Zheng, X.-Z. Lu, K.-Y. Chen, Y.-C. Zhou, and J.-R. Lin, "Pretrained domain-specific language model for natural language processing tasks in the AEC domain," *Computers in Industry,* vol. 142, p. 103733, 2022.

[37] R. Vunikili, H. Supriya, V. G. Marica, and O. Farri, "Clinical NER using Spanish BERT Embeddings," in *IberLEF@ SEPLN*, 2020, pp. 505-511.

[38] Y. Lu, R. Yang, X. Jiang, C. Yin, and X. Song, "A military named entity recognition method based on pre-training language model and BiLSTM-CRF," *Journal of Physics: Conference Series,* vol. 1693, no. 1, p. 012161, 2020.

[39] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems,* vol. 33, pp. 1877-1901, 2020.

[40] J. W. Rae, S. Borgeaud, T. Cai, K. Millican, J. Hoffmann, F. Song *et al.*, "Scaling language models: Methods, analysis & insights from training gopher," *arXiv preprint arXiv:.11446,* 2021.

[41] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts *et al.*, "Palm: Scaling language modeling with pathways," *arXiv preprint arXiv:.02311,* 2022.

[42] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford *et al.*, "Training compute-optimal large language models," *arXiv preprint arXiv:.15556,* 2022.

[43] OpenAI, "GPT-4 Technical Report," p. arXiv:2303.08774Accessed on: March 01, 2023. doi: 10.48550/arXiv.2303.08774 Available: https://ui.adsabs.harvard.edu/abs/2023arXiv230308774O

[44] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud *et al.*, "Emergent abilities of large language models," *arXiv preprint arXiv:.07682,* 2022.

[45] T. Kocmi, R. Bawden, O. Bojar, A. Dvorkovich, C. Federmann, M. Fishel *et al.*, "Findings of the 2022 conference on machine translation (WMT22)," in *Proceedings of the Seventh Conference on Machine Translation (WMT)*, 2022, pp. 1-45.

[46] N. Goyal, C. Gao, V. Chaudhary, P.-J. Chen, G. Wenzek, D. Ju *et al.*, "The flores-101 evaluation benchmark for low-resource and multilingual machine translation," *Transactions of the Association for Computational Linguistics,* vol. 10, pp. 522-538, 2022.

[47] R. Bawden, E. Bilinski, T. Lavergne, and S. Rosset, "DiaBLa: a corpus of bilingual spontaneous written dialogues for machine translation," *Language Resources Evaluation,* vol. 55, pp. 635-660, 2021.

[48] J. Liu, D. Shen, Y. Zhang, B. Dolan, L. Carin, and W. Chen, "What Makes Good In-Context Examples for GPT-3 ?," *arXiv preprint arXiv:.06804,* 2021.

[49] Z. Zhao, E. Wallace, S. Feng, D. Klein, and S. Singh, "Calibrate before use: Improving few-shot performance of language models," in *International Conference on Machine Learning*, 2021, pp. 12697-12706: PMLR.

[50] S. Wang, X. Sun, X. Li, R. Ouyang, F. Wu, T. Zhang *et al.*, "Gpt-ner: Named entity recognition via large language models," *arXiv preprint arXiv:.10428,* 2023.

[51] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix *et al.*, "Llama: Open and

efficient foundation language models," *arXiv preprint arXiv:.13971,* 2023.

[52] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin *et al.*, "Alpaca: A strong, replicable instruction-following model," *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu//03/13/alpaca. html,* vol. 3, no. 6, p. 7, 2023.

[53] P. Liu, Y. Guo, F. Wang, and G. Li, "Chinese named entity recognition: The state of the art," *Neurocomputing,* vol. 473, pp. 37-53, 2022.

[54] C. Wu, X. Wang, P. Wu, J. Wang, R. Jiang, M. Chen *et al.*, "Hybrid deep learning model for automating constraint modelling in advanced working packaging," *Automation in Construction,* vol. 127, p. 103733, 2021.

[55] Q. Cheng, T. Sun, W. Zhang, S. Wang, X. Liu, M. Zhang *et al.*, "Evaluating Hallucinations in Chinese Large Language Models," *arXiv preprint arXiv:.03368,* 2023.