AI Open 4 (2023) 111–119

Contents lists available at ScienceDirect

AI Open

journal homepage: www.keaipublishing.com/en/journals/ai-open

Joint span and token framework for few-shot named entity recognition

Wenlong Fang, Yongbin Liu *, Chunping Ouyang, Lin Ren, Jiale Li, Yaping Wan

School of Computer, University of South China, China

ARTICLE INFO

Keywords:
Few-shot learning
Named entity recognition
Meta learning
Natural language processing

ABSTRACT

Few-shot Named Entity Recognition (NER) is a challenging task that involves identifying new entity types using a limited number of labeled instances for training. Currently, the majority of Few-shot NER methods are based on span, which pay more attention to the boundary information of the spans as candidate entities and the entity-level information. However, these methods often overlook token-level semantic information, which can limit their effectiveness. To address this issue, we propose a novel Joint Span and Token (**JST**) framework that integrates both the boundary information of an entity and the semantic information of each token that comprises an entity. The **JST** framework employs span features to extract the boundary features of the entity and token features to extract the semantic features of each token. Additionally, to reduce the negative impact of the Other class, we introduce a method to separate named entities from the Other class in semantic space, which helps to improve the distinction between entities and the Other class. In addition, we used GPT to do data augmentation on the support sentences, generating similar sentences to the original ones. These sentences increase the diversity of the sample and the reliability of our model. Our experimental results on the Few-NERD[1] and SNIPS[2] datasets demonstrate that our model outperforms existing methods in terms of performance.

1. Introduction

Named Entity Recognition (NER) is a critical task that involves identifying entity positions from unstructured sentences and assigning them to predefined categories. Recently, deep learning-based approaches, such as Lample et al. (2016), Ma and Hovy (2016), Chiu and Nichols (2016), Gong et al. (2021) and Li et al. (2021b), have shown impressive performance on NER by leveraging large-scale, human-annotated datasets. However, in real-world applications, label annotations are often expensive and encountering new, unseen classes is common during training. In response, Few-shot NER has emerged as a promising approach for learning entity features of unseen types using only a few labeled instances. As a result, it has garnered considerable attention from the research community.

Most Few-shot NER methods rely on metric learning (Hou et al., 2020; Yang and Katiyar, 2020; Fritzler et al., 2019), which determines the class of a query by calculating its Euclidean distance from each class prototype. The prototypical network (Snell et al., 2017) is a popular and effective approach for Few-shot NER, where prototypes are learned for each predefined entity class, and the query samples are classified based on the closest prototype. Recently, researchers have proposed span-level metric learning methods for Few-shot NER (Yu et al., 2021; Wang et al., 2021), which utilize span information to extract entity features. In this approach, the category of entities in the query is determined by measuring the similarity between the spans in the support and query sets.

The span-level metric learning methods have certain limitations. Firstly, they focus primarily on the boundary information of spans as potential entities and entity-level information, while token-level semantic information is often neglected. For instance, when the entity "*Southern Sudan Autonomous Region*" is transformed into spans, only the head and tail token information is retained, leaving out other vital token information such as "*Sudan Autonomous*", as depicted in Fig. 1(a). Consequently, the semantic features provided by support examples may not be adequately learned. Secondly, the predefined entity classes and the other class have distributional biases. A large proportion of the tokens of the other class exert complex effects on the learning of predefined entity class prototypes. Moreover, in cases where a token is classified as the other class type during training, the same token could belong to a valid entity class during testing, as exemplified in Fig. 1(b). These issues exacerbate the adverse impacts of the other class on learning entity prototypes.

To address the aforementioned issues, we propose a joint span and token framework for few-shot named entity recognition. This

* Corresponding author.
E-mail address: yongbinliu03@gmail.com (Y. Liu).

[1] https://ningding97.github.io/fewnerd/.

[2] https://github.com/AtmaHou/FewShotTagging.

https://doi.org/10.1016/j.aiopen.2023.08.009
Received 14 July 2023; Received in revised form 6 August 2023; Accepted 20 August 2023
Available online 4 September 2023

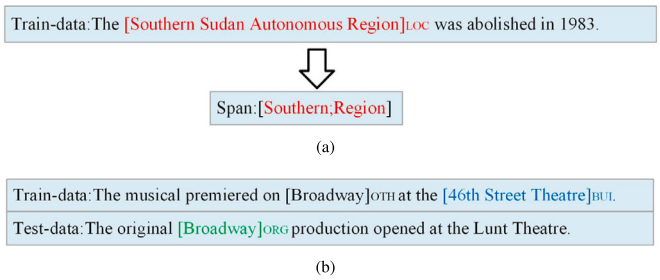


Fig. 1. (a) The span-based methods transform an entity into a span by utilizing its start and end words as its representation. (b) Tokens may be classified differently in two sentences, posing a challenge for entity classification. "LOC" represents "Location", "ORG" represents "Organization", "BUI" represents "Building", and "OTH" represents the other class.

framework prioritizes both the boundary information of an entity and the semantic information of its constituent tokens. Firstly, we extract the boundary features of each entity using intra-sentence and inter-sentence spans to reinforce each other. Secondly, we extract the features of each token, including its semantic features, to calibrate the prototypes. The relevance of tokens in the support and query is computed to enhance the feature expression of the support. Thirdly, we fuse the span and token features to learn a new joint feature vector matrix, to structure the prototypes. Finally, we integrate the aforementioned modules to consider both the boundary information of an entity and the semantic information of its constituent tokens.

To mitigate the difficulty of distinguishing between the other class and entity classes caused by a large number of instances of the other class with miscellaneous semantics, we propose an "Other Class Attention" approach that separates the other class from the entities in the embedding space. This is achieved by calculating a similarity score between the other class and entities, which determines the extent to which features from the other class affect the entity prototypes. We then reweight these features to move the other class farther away from the entities in the embedding space.

Few-shot named entity recognition poses the challenge of limited training samples, leading to overfitting or poor generalization of models to new data. To augment the data for few-shot named entity recognition, we propose utilizing the GPT 3.5 API. By setting a specific task and providing relevant samples, we can leverage GPT's capabilities to generate additional sentences and entities that we require as supplementary data. We input the sentences and entities from the support set into GPT, and it generates sentences containing the specified entity categories based on the provided input. These generated sentences are then incorporated back into the support set, enriching the training samples with more diverse examples. This data augmentation process enhances the model's ability to generalize and improves its performance by introducing additional variations and contexts.

Our main contributions are as follows:

- We propose a joint span and token framework that captures both the semantic features of tokens and the entity-level information.
- We propose a method to mitigate the negative impacts of the other class by obtaining other class features that are farther away from the entities in the embedding space.
- We utilized the GPT 3.5 API for data augmentation to increase sample diversity and enhance model reliability in named entity recognition.
- We evaluate our model on the Few-NERD and SNIPS datasets under various few-shot settings, and our experimental results demonstrate that our model outperforms state-of-the-art models, making it more reliable and transferable.

2. Related work

Now NER is getting more and more attention and can be used in a variety of application scenarios (Wen et al., 2021; Li et al., 2021a), Few-NERD (Ding et al., 2021) proposed a dataset and framework for few-shot NER applicability, and the results of few-shot NER are getting better with the combination of meta-learning and few-shot.

Many few-shot problems (Lin et al., 2022) can be combined with meta-learning (Hochreiter et al., 2001) and achieve better results. The central idea of meta-learning is to help models quickly adapt to new tasks. Traditional meta-learning, such as Prototype network (Snell et al., 2017). Matching networks (Vinyals et al., 2016) and HATT (Gao et al., 2019) proposes a hybrid attention based on a prototype network and a feature-level and instance-level based attention, effectively reducing the noise. Some papers (Ravi and Larochelle, 2016; Elsken et al., 2020; Vanschoren, 2018; Munkhdalai and Yu, 2017) propose meta-learning based on parameter optimization, hoping to adapt to new tasks quickly. MAML (Finn et al., 2017) learns an initial parameter and then fine-tunes the task based on this parameter. Matching Net (MN)BERT (Hou et al., 2020) uses the matching network for few-shot NER.

The first class of Few-shot NER is based on the token: ProtoBERT (Fritzler et al., 2019), used Prototype network for NER. NNShot (Yang and Katiyar, 2020) uses the nearest neighbor algorithm for classification, and StructShot (Yang and Katiyar, 2020) adds a CRF as a decoder with NNshot. CONTAINER (Das et al., 2021) uses Gaussian embedding and introduces contrast learning to improve the model's ability. TransferBERT (Hou et al., 2020) is based on one application of fine-tuning to bert. ConVEx (Henderson and Vulić, 2020) is a fine-tuning-based model to get the final few-shot sequence tags. L-TapNet-CDT (Hou et al., 2020) proposes to use pair-wise embedding, label semantics, and CDT transition mechanism for NER.

The second class is based on span: ESD (Wang et al., 2021) uses multiple attention, learns features from different categories. Decomposed (Ma et al., 2022) uses meta-learning to handle span detection and entity classification problems. ANL (Athiwaratkun et al., 2020) and Ma2021 (Ma et al., 2021) view the sequence criteria problem as a reading comprehension problem. Template-Based Bert (Cui et al., 2021) turn NER tasks into template-based scoring tasks. Retriever (Yu et al., 2021) classifies instances based on their similarity to support.

3. Methodology

This section provides a comprehensive overview of our model's architecture, as depicted in Fig. 2. Our model is designed to address the challenges of few-shot named entity recognition (NER) by incorporating multiple modules to capture essential entity features.

(1) Span Module: We utilize the span module to comprehend span-level information. Through Intra Span Attention, we identify significant spans within the same sentence. Subsequently, Other Class Attention is applied to support spans, creating a distinct separation between Other Classes and Entity Class features in the vector space. Cross Sentence Span Attention is then employed to bring the support and query spans closer together in the vector space, refining feature representation. (2) Token Module: Our token module is responsible for acquiring token-level information. It constructs prototypes adaptable to queries by computing token similarity between support and query. (3) Hybrid Module: Incorporating both token and span details, the hybrid module constructs prototypes by approximating token and span features, facilitating a mutually enhanced expression. (4) Joint Span and Token Probability Distribution: We amalgamate the probability distributions from the three modules to generate the final probability distribution of our model, enabling accurate entity classification. (5) GPT Data Augmentation: We used GPT to augment the data with sentences from support, improving the transferability of our model and the diversity of our samples.

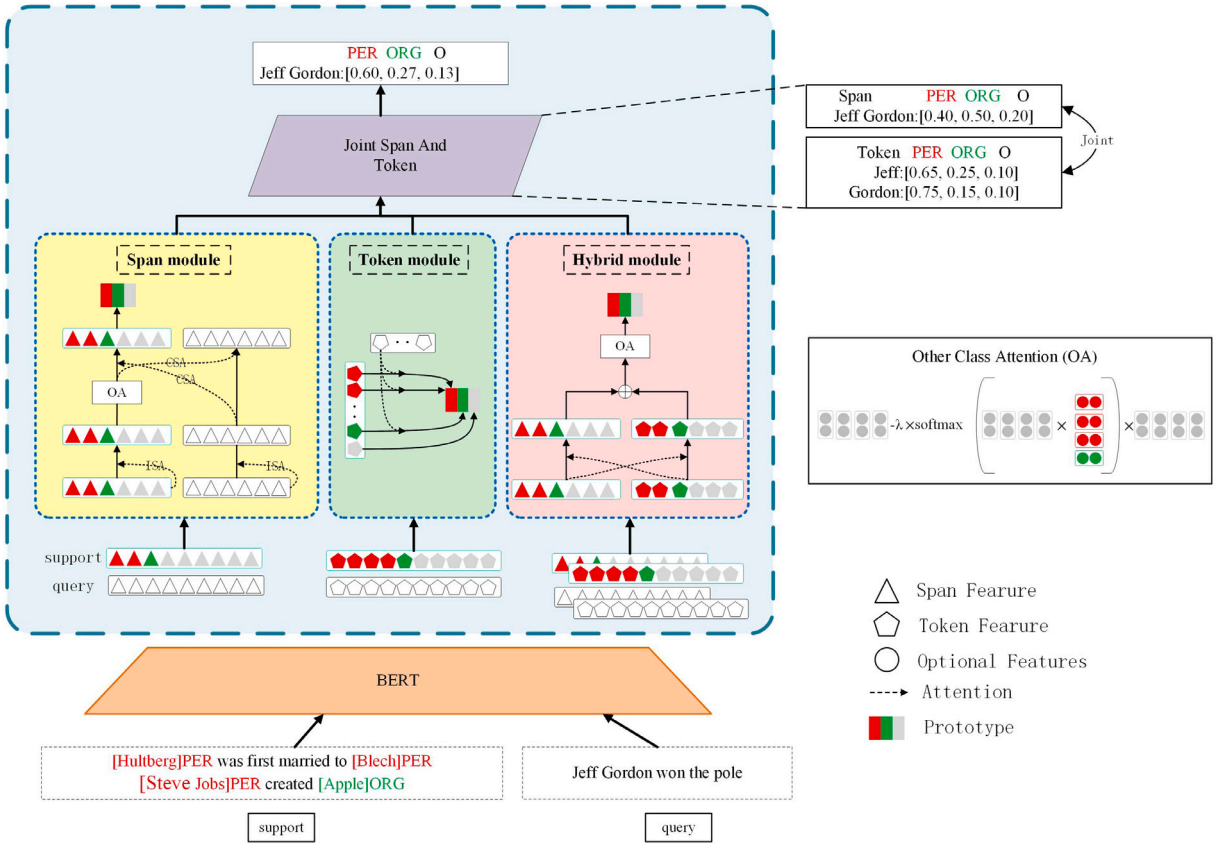


Fig. 2. Joint span and token framework: The triangles in the figure represent span features and the pentagons represent token features. Circles can be used as both span features and token features, depending on the context in which they are used. In Span module it represents span features, in Hybrid module it represents token and span features. Different colors represent different entity classes, gray represents other class, and no color represents an unseen class from the query. The color red corresponds to the entity class "person", while the color green signifies the entity class "organization".

We joint span and token for Few-shot NER, our method can learn both the boundary information of the entity and the information of each token that constitutes the entity. We also propose a method to keep other class away from entities. This method effectively alleviates the negative effects of other class.

3.1. Span module

In this section, we describe the span-based NER of our model. We are inspired by ESD (Wang et al., 2021), we use BERT (Devlin et al., 2018) for feature extraction of sentences and take the last layer of the output of BERT as the feature of our sentences. Each sentence has N words, $\mathbf{W} = (w_1, w_2, \ldots, w_N)$, and the corresponding feature for each word is $\{x_1, x_2, \ldots, x_N\}$, x represents the output of each word through BERT, and N represents the nth word in the sentence. A span $s = (l, r)$, l and r represent the position of the begin and end of the entity in the sentence, respectively, and the feature expression of the span is $\mathbf{s}_{(l,r)} = [\mathbf{x_l}; \mathbf{x_r}]$.

3.1.1. Intra Span Attention

To capture salient span features within a sentence and enhance their representations, we introduce the concept of **Intra Span Attention(ISA)**. The goal is to make the model pay closer attention to relevant span features in the same sentence. By considering all spans within the sentence and leveraging mutual enhancement, the meaning of a particular span can be better inferred. Intra Span Attention enables the model to attend to and utilize information from all spans in a sentence, facilitating a more comprehensive understanding of the context. As shown in the span module in Fig. 2, we use ISA To capture salient span features for spans in the same sentence. We define the expression of span in a sentence as $\mathbf{S} = [s_1, s_2, \ldots, s_B]$, the span in the sentence as s_i, i means the ith span in the sentence, B is the number of spans in this sentence. We get $\tilde{\mathbf{s}}_\mathbf{i}$ by the following method

$$\tilde{\mathbf{s}}_\mathbf{i} = \sum_{j=1}^{B} \alpha_\mathbf{i}^j \mathbf{s_j} \tag{1}$$

$$\alpha_\mathbf{i} = \mathrm{softmax}\left(\mathbf{s_i}\mathbf{S}^\mathbf{T}\right) \tag{2}$$

to make it easier to understand, we define this operation as ϕ,

$$\tilde{\mathbf{s}}_\mathbf{i} = \phi\left(\mathbf{s_i}, \mathbf{S}\right) \tag{3}$$

3.1.2. Other class

The presence of numerous other class spans, each with miscellaneous semantics, poses challenges in distinguishing them from entity classes during classification. This difficulty can negatively impact entity classification performance. To address this issue, we propose a solution called **Other Class Attention(OA)**. The purpose of OA is to create a separation between other class spans and entity classes in the embedding space. By implementing OA, we aim to improve the model's ability to discriminate between entity and other class spans, reducing the interference caused by the presence of diverse other class spans. As shown in the Other Class Attention in Fig. 2, we employ OA to

create separation between the entity class and the other class within the vector space. This is achieved by diminishing the influence of the other class instances that are in close proximity to the entity's feature representation. OA effectively enhances the discriminative capacity of our model by emphasizing the distinctiveness of entity and other class features. We take the entity span $E = [E_1, E_2, \ldots, E_n]$ and other class span $O = [O_1, O_2, \ldots, O_m]$ in spans of support for attention calculation. E is the matrix of all entity spans in support, E_n represents the nth row of E, and similarly, O is the matrix of all other class spans in support, O_m represents the mth row of O. The other class features that are further away from the entities are obtained by

$$\alpha_{\mathrm{o}} = \mathrm{softmax}\left(\max\left(O\mathrm{E}^{\mathrm{T}}\right)\right) \tag{4}$$

$$\check{\mathrm{O}} = O - \lambda\alpha_{\mathrm{o}}O \tag{5}$$

λ is the hyper-parameter of model reliability, which is obtained by multiple episodes. Finally, we replace the span of other class in support with $\check{\mathrm{O}}$.

3.1.3. Cross Sentence Span Attention

To tackle the issue of span matching between queries and support, we introduce a technique called **Cross Sentence Span Attention(CSA)**. This approach enhances the information of both the query span and support span by integrating their representations. By doing so, we bring the query span and support span closer in vector space, facilitating the construction of prototypes that are better adapted to the query span. This attention mechanism plays a crucial role in aligning the features of query and support spans, improving the model's ability to recognize and classify named entities accurately across sentences. As shown in the span module in Fig. 2, we harness the interplay between support and query to enhance their feature representations through mutual attention. This dynamic interaction facilitates the refinement of both support and query features. Subsequently, we leverage the augmented support features to construct prototypes that are finely tailored to the characteristics of the query. The set of all spans of support constitutes a matrix $\tilde{\mathbf{S}}$, $\tilde{\mathbf{S}} = [\tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_n]$, with $\tilde{s}_n$ representing the nth row of $\tilde{\mathbf{S}}$. Similarly, we define the matrix of all spans in Query as $\tilde{\mathbf{Q}}$, with $\tilde{q}_m$ representing the mth row for $\tilde{\mathbf{Q}}$. Then the initial cross-sentence expressions are obtained by $\check{\mathbf{s}}_{\mathbf{n}} = \phi\left(\tilde{\mathbf{s}}_{\mathbf{n}}, \tilde{\mathbf{Q}}\right)$ and $\check{\mathbf{q}}_{\mathbf{m}} = \phi\left(\tilde{\mathbf{q}}_{\mathbf{m}}, \tilde{\mathbf{S}}\right)$.

Indeed, the prototype network is a powerful and straightforward approach for few-shot classification. It operates by computing class prototypes, which are the averages of all samples belonging to the same class. Each class is represented by its respective prototype. To classify an entity, the Euclidean distance between the entity and each prototype is calculated. Based on these distances, the probability that the entity belongs to each class is determined. The class with the highest probability is then selected as the predicted class for the entity. We construct the prototype of each class from all the spans of that class in support. We get each prototype by by the following method:

$$p_i^s = \frac{1}{|c_i|}\sum_{j=1}^{|c_i|}\left(\check{\mathbf{s}}_{ci,j}\right) \tag{6}$$

where j is the jth spans in c_i class, c_i represents the ith class in the predefined classes of support set, $|c_i|$ stands for the number of spans in c_i class. We predict $\check{\mathbf{q}}_{\mathbf{m}}$ to be the c_i type in the support set with probability:

$$g^s\left(y = c_i \mid \check{\mathbf{q}}_{\mathbf{m}}\right) = \frac{\exp\left(-d\left(\left(\check{\mathbf{q}}_{\mathbf{m}}\right), p_i^s\right)\right)}{\sum_{j=1}^{|C|}\exp\left(-d\left(\left(\check{\mathbf{q}}_{\mathbf{m}}\right), p_j^s\right)\right)} \tag{7}$$

where d is the euclidean distance, $|C|$ stands for the number of classes in the class set. During training, we use cross-entropy as this module loss function,

$$\mathcal{L}_1 = -\frac{1}{n}\sum_{m=1}^{n}\log g^s\left(y_m^* \mid q_m\right) \tag{8}$$

where n is the number of spans in query, y_m^* is the true label of q_m.

3.2. Token module

In this section, we present the token-based NER module, designed to focus on the token features in the support set that are most relevant to the query entities. To achieve this, we introduce token-level attention. By computing a similarity score matrix between the tokens in the support and query, we can identify the support features that are most pertinent to the query entity. With this information, we construct prototypes that are tailored to the specific query entity, enhancing the model's ability to recognize and classify the query entity accurately. The token-based NER module effectively leverages token-level attention to adapt the prototypes to each query entity, contributing to improved performance in few-shot NER tasks. We obtain the feature vector x for each token in sentences by BERT encoding. Our model calculates the correlation formula as follows:

$$\beta_j = \frac{\exp\left(-d\left(x_q, x_{c_{i,j}}\right)\right)}{\sum_{k=1}^{|c_i|}\exp\left(-d\left(x_q, x_{c_{i,k}}\right)\right)} \tag{9}$$

where x_q represents the token of query, $x_{c_{i,j}}$ represents the jth token of support in c_i class, $|c_i|$ stands for the number of tokens in c_i class, d is the euclidean distance. Then, for each query sample x_q, the prototypes of classes is defined as:

$$p_i^t = \frac{1}{|c_i|}\sum_{j=1}^{|c_i|}\left(\beta_j x_j\right) \tag{10}$$

We predict x_q to be the c_i type in the support set with probability:

$$g^t\left(y = c_i \mid x_q\right) = \frac{\exp\left(-d\left(\left(x_q\right), p_i^t\right)\right)}{\sum_{j=1}^{|C|}\exp\left(-d\left(\left(x_q\right), p_j^t\right)\right)} \tag{11}$$

we use cross-entropy as this module loss function,

$$\mathcal{L}_2 = -\frac{1}{n}\sum_{m=1}^{n}\log g^t\left(y_q^* \mid x_q\right) \tag{12}$$

where n is the number of tokens in query, y_q^* is the true label of x_q.

3.3. Hybrid span and token feature module

This section mainly describes the token and span hybrid module. To focus on the features that are most relevant to the entity, we employ a hybrid representation that enhances entity features using both span features and token features. The goal is to create a balance between the span features that are closely related to the tokens and the token features that are closely related to the spans. To achieve this, we perform attention computation between span features and token features, allowing them to mutually reinforce each other's expression. After this attention-based computation, we combine the weighted span features and token features together. This approach helps us capture essential information from both span and token levels and construct prototypes that contain both marking information and span information, which effectively improving the model's ability to recognize and extract relevant features for named entities. We collect all spans to form a matrix $\mathbf{S} = [s_1, s_2, \ldots, s_n]$, n representing the nth row of $\mathbf{S}$. We collect all tokens to form a matrix $\mathbf{T} = [t_1, t_2, \ldots, t_m]$, m representing the mth row of $\mathbf{T}$. We take the maximum value of n or m and assign it to m. We define m as the maximum number of rows of $\mathbf{S}$ and $\mathbf{T}$. The matrix with the smaller number of rows is filled with zeros until this matrix has m rows. We obtain the fused features by mixing the features of each span and the features of each word. We construct the feature matrix with span and token mixing. The prototype constructed by this matrix contains both span-level features and token-level features. We get the span feature closer to the token expression $\mathbf{s'}_{\mathbf{m}} = \phi\left(\mathbf{s}_{\mathbf{m}}, \mathbf{T}\right)$. We get a token feature closer to the span expression $\mathbf{t'}_{\mathbf{m}} = \phi\left(\mathbf{t}_{\mathbf{m}}, \mathbf{S}\right)$. Finally, we

add the span features and token features to obtain a token-level vector matrix that combines span and token features.

$$\hat{t}_m = t_m + \tau(\mathbf{t'_m} + \mathbf{s'_m}) \tag{13}$$

where τ is the hyper-parameter of the model. By Eq. (13) we get $\hat{t}_{sm}$ of support and $\hat{t}_{qm}$ of query. Meanwhile, in order to reduce the noise brought by other class, we use other class attention to pull away the Euclidean distance between other class and entity classes in embedding space. We take the entity and other class in $\hat{t}_{sm}$ to constitute O_{sm} and E_{sm}. We put O_{sm} and E_{sm} into Eqs. (4) and Eqs. (5) to get $\hat{O}_{sm}$. We use $\hat{O}_{sm}$ to replace the other class tokens in support $\hat{t}_{sm}$. The prototype in our model is defined as:

$$p_i^m = \frac{1}{|c_i|}\sum_{j=1}^{|c_i|}\left(\hat{t}_{ci,j}\right) \quad \hat{t}_{ci,j} \in \hat{t}_{sm} \tag{14}$$

where $\hat{t}_{ci,j}$ is the jth token in c_i type. We predict $\hat{t}_{qm}$ to be the c_i type in the support set with probability:

$$g^h\left(y = c_i \mid \hat{t}_{qm}\right) = \frac{\exp\left(-d\left(\hat{t}_{qm}, p_i^m\right)\right)}{\sum_{j=1}^{|C|}\exp\left(-d\left(\hat{t}_{qm}, p_j^m\right)\right)} \tag{15}$$

we use cross-entropy as this module loss function,

$$\mathcal{L}_3 = -\frac{1}{n}\sum_{m=1}^{n}\log g^h\left(y_{qm}^* \mid \hat{t}_{qm}\right) \tag{16}$$

where n is the number of tokens in query, y_{qm}^* is the true label of $\hat{t}_{qm}$.

3.4. Joint span and token probability distribution

In Few-shot NER, the limited labeled data in each support set necessitates our model to identify unseen classes with only a few labeled examples. This demands a highly transferable model capable of learning comprehensive entity features. To achieve this, we utilize the span module to capture entity boundary and entity-level information. Additionally, the token module and hybrid module are employed to extract semantic information from each token that constitutes an entity. By combining the information from these three modules, we make a joint determination of the entity's class. This integrated approach enables our model to effectively learn and utilize diverse features, enhancing its ability to recognize and classify named entities with few labeled examples, thus improving transferability. This is achieved through the joint of probabilities from these three modules for entity classification. The probabilities are combined by assigning weights to each module's contribution. To unify the probability from the token module and the probability from the span module, we assign weights to the probabilities associated with the entity's initial and final tokens. These weighted probabilities yield the probability corresponding to the span of the specific entity. This comprehensive approach effectively fuses the insights from token and span information, enhancing the accuracy and robustness of our entity classification process. The probability distribution of the joint token level g^{J_t} is obtained by g^t and g^h.

$$g^{J_t}\left(y = c_i \mid x\right) = g^t\left(y = c_i \mid x\right) + g^h\left(y = c_i \mid x\right) \tag{17}$$

We obtain the joint span and token final probability distribution by g^{J_t} and g^s

$$\begin{aligned} &g^{J_{st}}\left(y = c_i \mid q_m\right) = g^s\left(y = c_i \mid q_m\right) + \\ &\sigma\left[g^{J_t}\left(y = c_i \mid x_l\right) + g^{J_t}\left(y = c_i \mid x_r\right)\right] \end{aligned} \tag{18}$$

where l and r represent start position and end position of span q_m, x_i represent the ith token of the sentence of q_m. σ is the hyper-parameter of model reliability, which is obtained by multiple episodes.

3.5. GPT data augmentation

In few-shot learning, the model is provided with only a limited number of samples, expecting it to extract sufficient feature information for subsequent classification. To alleviate this limitation, we utilize GPT for data augmentation. Specifically, we input support samples into GPT, allowing it to learn the entity features present in the samples and generate similar samples based on the input. These generated samples are then incorporated back into the support set and used to train the model, addressing the challenge posed by few-shot.

3.5.1. Description of the task

In Fig. 3, we provide a concrete example of using the GPT-3.5 API for data augmentation. The process involves three steps.

First, we generate entities based on the provided entities and entity classes. GPT identifies and learns the features of the entities based on the examples provided, preparing for the next step.

Next, we generate a sentence containing the entities generated in the previous step, based on the provided sentences. This step focuses on data augmentation by generating sentences that include the specified entity class. Since few-shot entity recognition requires the specified entity class for each input, the generated sentence needs to align with the provided entity class.

Finally, we add some requirements for the specified sentences, ensuring they are less than 50 words long and answer the questions in order. This step ensures that multiple questions are answered sequentially and that the generated sentences adhere to the length constraint.

Due to the limitations of the GPT-3.5 API, there are constraints on the length of the prompt and the number of tokens. To overcome this, data augmentation can be performed iteratively by asking a certain number of questions at a time and repeating the three steps multiple times to cover the desired data augmentation scope.

3.5.2. Sample tasks

In this section, we utilize GPT with provided task samples and define specific output formats to ensure consistency and facilitate data processing. By presenting GPT with samples of our task and limiting the output format, we aim to improve GPT's understanding of our specific task requirements. This approach helps GPT generate more accurate and relevant output for data augmentation, aligning with our intended format and making subsequent data processing easier. Fig. 3 showcases a sample of the task setup, highlighting the steps we take to guide GPT and achieve desirable output results for data augmentation.

3.5.3. Input and output of the task

In our data augmentation process, we input the sentence to be augmented into GPT and extract each entity and its corresponding entity class from the sentence. The number of questions in this part of the input is the total number of sentences in the support data in each round of the few-shot NER. Following the specified format, we input this information into GPT, which generates entities based on the provided entity classes and generates similar sentences based on the input entities and sentences. It is important that the output sentence from GPT matches the entity class of the input sentence. If the generated sentence contains entity classes that were not provided, it cannot be used for data augmentation.

The generated entities and sentences, which correspond to the specified entity classes, serve as reliable similar samples for the task of few-shot named entity recognition. They are not negative samples that would adversely affect the model's performance. For example, in Fig. 3, the input sentence is "*Ironically, the team left Connie Mack Field for Bradenton, after the Milwaukee Braves arrived in West Palm Beach*". The entities extracted from the sentence are "*Connie Mack Field, Bradenton, Milwaukee Braves, West Palm Beach*" with the corresponding entity classes being "*organization, location, organization, location*". The output

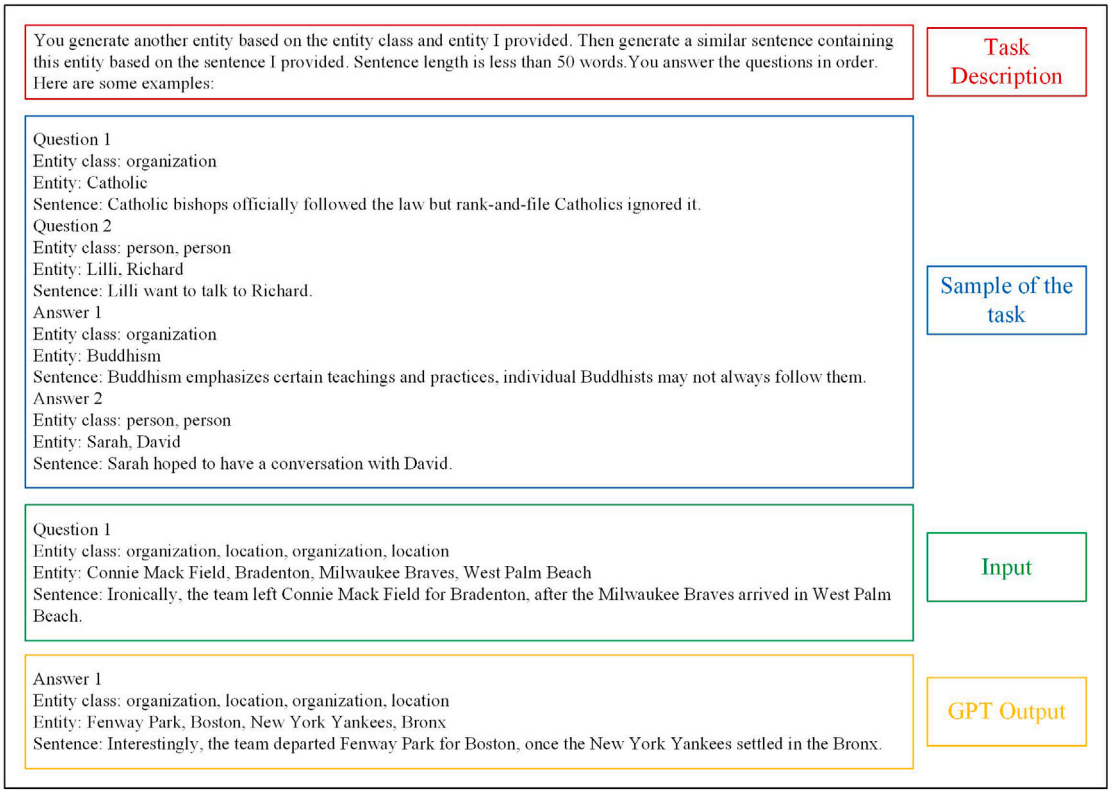


Fig. 3. Example of GPT Data Augmentation prompt. We need to generate entities of the same class and sentences based on the given sentences, entities and entity categories. The prompt consists of three parts: (1) Task Description: It is surrounded by a red rectangle and indicates that the current task of the GPT-3.5 model is to identify entity categories using linguistic knowledge and to generate sentences with entities of the same category. (2) Few-shots Sample of the task:It is surrounded by a blue rectangle and provides few-shots examples for the GPT-3.5 model to refer to. (3) Input Sentence: It is surrounded by a green rectangle, which represents the input sentence. (4) Output sentence:It is surrounded by an orange rectangle indicating the output of the GPT-3.5 model.

sentence generated by GPT, "*Interestingly, the team departed Fenway Park for Boston, once the New York Yankees settled in the Bronx*". contains the "*organization, location*" entity class. With the output entities and entity classes, we obtain labeled sentences for data augmentation. The label of the output sentence in Fig. 3 is "*O O O O O ORG ORG O O LOC O O O ORG ORG ORG O O O LOC O*" where "O" represents the non-entity class, "ORG" represents the organization entity class and "LOC" represents the location entity class.

Using data augmentation in this way can significantly improve the performance of our model in few-shot named entity recognition tasks.

4. Experiments

In this section, we will describe our model's experimental setup, the dataset used, and the parameter settings of the experiment. We will also compare our model with other Few-shot NER models from recent years. We further investigate the effects of different parts of our model.

4.1. Experiments datasets

Our model has mainly experimented on two datasets. Few-NERD (Ding et al., 2021) is a large-scale Few-shot NER dataset containing 188,238 Wikipedia-based sentences and 4,601,160 words with two-level tags. The labels of the entities include eight coarse-grained and 66 fine-grained. Few-NERD is divided into two main tasks: FEW-NERD(INTER) and FEW-NERD(INTRA). The entity categories of FEW-NERD(INTER) are derived from the same coarse-grained but different fine-grained. The entity categories of FEW-NERD(INTRA) are derived from the different coarse-grained. N way - k shot means that there are N categories of entities and K instances of each category.SNIPS (Hou et al., 2020) is also a Few-shot NER dataset, mainly consisting of data from 7 domains. One domain is randomly selected as a test each time, another is selected as validation, and the remaining 5 are used as training. SNIPS does not specify the number of entity types because the number of entity classes in different training sets is not necessarily the same for each training.

4.2. Parameter settings

Following previous methods (Wang et al., 2021; Ma et al., 2022), we use BERT-base-uncased (Devlin et al., 2018) as the encoder, the hidden dimension of BERT is 768, and the learning rate of BERT is 1e−4 with 1000 warmup step. We use a batch size of 1, a maximum sequence length of 60, and a dropout rate (Srivastava et al., 2014) probability of 0.1. The maximum span length is set to 8. The σ for joint span and the token is 0.5, and the λ of other class attention is 0.25. We use Adam (Kingma and Ba, 2014) as our optimizers. Our model has 10,000 iterations for training, 1000 iterations for validation, and 500 iterations for testing.

4.3. Baselines

ProtoBERT (Snell et al., 2017), which uses the prototype network for few-shot NER. **NNshot** (Yang and Katiyar, 2020) uses the nearest neighbor algorithm for classification. **Structshot** (Yang and Katiyar, 2020), the same structure as NNshot but with the addition of CRF. **L-TapNet-CDT** (Hou et al., 2020) proposes to use pair-wise embedding, label semantics, and CDT transition mechanisms to improve model performance. **ESD** (Wang et al., 2021) uses multiple attention to learn features of the same and different sentences. **Container** (Das et al., 2021) introduces contrast learning and Gaussian embedding to learn features of sentences. **Decomposed** (Ma et al., 2022) uses

Table 1
F1 scores with standard deviations on Few-NERD for both inter and intra settings. The best results are in **boldface**.

Model	intra				inter			
	5-1	5-5	10-1	10-5	5-1	5-5	10-1	10-5
ProtoBERT	23.45	41.93	19.76	34.61	44.44	58.80	39.09	53.97
NNShot	31.01	35.74	21.88	27.67	54.29	50.56	46.98	50.00
StructShot	35.92	38.83	25.35	26.39	57.33	57.16	49.46	49.39
CONTAINER	40.40	53.70	33.84	47.49	55.95	61.83	48.35	57.12
ESD	41.44	55.62	32.29	42.92	66.03	75.32	59.95	67.91
Decomposed	52.04	63.23	43.50	**56.74**	68.77	71.62	63.26	68.32
JST	53.06	63.84	44.47	55.43	70.52	78.94	64.21	74.17
JST-GPT	**55.74**	**64.23**	**48.03**	56.32	**75.37**	**79.86**	**69.86**	**75.23**

meta-learning to handle span detection and classification. **Ma2021** (Ma et al., 2021) views the sequence criterion problem as a reading comprehension problem. **ConVEx** (Henderson and Vulić, 2020) base fine-tuning to obtain the few-shot sequence tags. **Retriever** (Yu et al., 2021) classifies instances based on their similarity to support.

4.4. Evaluation metrics

For the evaluation metrics of Few-NERD dataset, we use the evaluation metrics of Few-NERD (Ding et al., 2021). Using the data from the test set, we calculate the precision (P), recall (R), and micro F1-score (F1) as the evaluation metrics of the model. For SNIPS dataset, we refer to SNIPS (Hou et al., 2020), where we first calculate the F1 value for each task in the test set, then average the F1 of the tasks, and use the averaged F1 as the evaluation metric for each test set.

4.5. Experimental results and analysis

In this section, we present a comparison between our model and the baseline on different datasets. Table 1 displays the results for the Few-NERD dataset, where our model without GPT outperforms previous methods on both INTER and INTRA, with significant improvements in the INTER setting. For example, in the 5-way 5-shot scenario on the INTER, our model achieves a 7.32% higher performance than the Decomposed baseline. Similar improvements are observed in other scenarios as well. Table 2 shows the results for the SNIPS dataset, where our model without GPT surpasses L-TapNet+CDT by 5.68% on average in the 1-shot setting, and outperforms ESD by 5.84% on average in the 5-shot setting. These results demonstrate the reliability and superiority of our model.

Furthermore, our model exhibits even better performance when GPT is employed for data augmentation. According to the results in Tables 1 and 2, in the 5-shot scenario, there is an average improvement of 0.815% in the Few-NERD dataset and 0.64% in the SNIPS dataset. In the 1-shot scenario, there is an average improvement of 4.185% in the Few-NERD dataset and 1.28% in the SNIPS dataset. These results highlight the advantages of incorporating GPT for data augmentation, which increases sample diversity and enhances model reliability, particularly in the 1-shot scenario.

4.5.1. Ablation study

In order to evaluate the individual contributions of different components in our model, we conducted an ablation study on the Few-NERD (INTER) dataset for 5way-1shot and 5shot scenarios. The results are presented in Table 3. (1) without GPT Data Augmentation, the F1 scores decreased by 4.85 and 0.92 for 5way-1shot and 5shot respectively. This highlights the effectiveness of the GPT Data Augmentation. (2) without Other Class Attention and GPT, the F1 scores dropped by 6.94 and 3.43. This emphasizes the significance of the Other Class Attention in separating entity features from other classes in the embedding space. (3) To assess the effectiveness of each module, we conducted performance tests individually. Notably, the performance of the span module alone exhibited promising results. However, even with its commendable performance, it fell short of the collective efficacy achieved by our comprehensive model. This suggests that while each module has intrinsic value, their combined integration within our model yields superior performance compared to their isolated implementations. The synergy achieved through the joint utilization of all three modules contributes significantly to the model's overall enhanced performance.

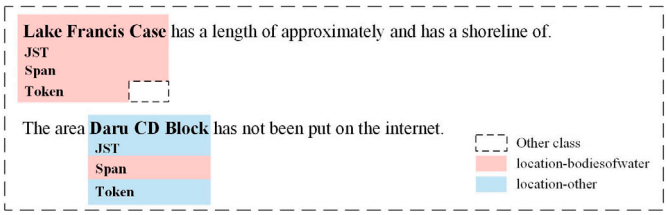
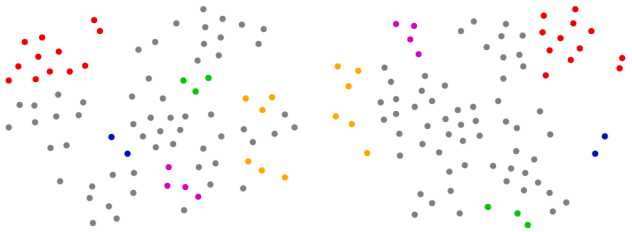


Fig. 4. A case of the span-based, token-based, and JST in the few-shot NER.



(a) w/o OA (b) w OA

Fig. 5. t-SNE visualization of token representations for entity typing on Few-NERD(INTER). Different colors represent different entity classes, where gray represents other class.

4.5.2. Impact of joint span and token

To validate our joint span and token model for Few-shot NER, we compared it with separate span and token modules using the FEW-NERD (INTER) dataset. Fig. 4 illustrates our analysis. In the first sentence of Fig. 4, the span module accurately captures entity boundaries, resulting in correct predictions. However, the token module struggles to identify boundaries, leading to incorrect outcomes despite accurate entity classification. This emphasizes the span module's proficiency in boundary detection, while the token module falls short. In the second sentence of Fig. 4, the span module focuses on boundaries but misclassifies entities, leading to incorrect predictions. In contrast, the token module effectively learns semantic features, yielding accurate results. This highlights the token module's ability to grasp the semantic information encoded in entity tokens. By combining the strengths of both modules, our joint span and token model incorporates boundary information and semantic understanding concurrently. This enhances the efficiency of entity recognition. Our model demonstrates its effectiveness in recognizing entities by considering both boundary features and semantic properties simultaneously.

4.5.3. Impact of other class attention

We examined the impact of our Other Class Attention (OA) on our model using the FEW-NERD (INTER) dataset. The results, presented in Table 4, indicate significant improvements when incorporating OA. Specifically, the span module achieved a 1.07 higher F1 score, while the hybrid module showed a more substantial increase of 1.59. To visually illustrate the influence of OA, we employed t-SNE (Van der Maaten and Hinton, 2008) for dimensionality reduction. Fig. 5 demonstrates that before using OA, instances from the other class were scattered in proximity to the entities. However, after implementing OA, a noticeable separation can be observed, with the other class being slightly pushed away from the entities in the semantic space. These findings confirm

Table 2
F1 scores with standard deviations on 7 domains of SNIPS. The best results are in **boldface**.

	Models	We	Mu	Pi	Bo	Se	Re	Cr	Avg.
1-shot	TransferBERT	55.82	38.01	45.65	31.63	21.96	41.79	38.53	39.06
	MN+BERT	21.74	10.68	39.71	58.15	24.21	32.88	69.66	36.72
	ProtoBERT	46.72	40.07	50.78	68.73	60.81	55.58	67.67	55.77
	Ma2021	–	–	–	–	–	–	–	69.3
	L-TapNet+CDT	71.53	60.56	66.27	84.54	76.27	70.79	62.89	70.41
	ESD	76.77	55.70	66.27	70.51	68.42	72.04	78.38	69.72
	JST	80.13	62.46	75.20	80.85	76.87	78.12	79.02	76.09
	JST-GPT	81.75	64.94	75.71	81.38	77.91	79.47	80.43	**77.37**
5-shot	TransferBERT	59.41	42.00	46.07	20.74	28.20	67.75	58.61	46.11
	MN+BERT	36.67	33.67	52.60	69.09	38.42	33.28	72.10	47.98
	ProtoBERT	67.82	55.99	46.02	72.17	73.59	60.18	66.89	63.24
	Retriever	82.95	61.74	71.75	81.65	73.10	79.54	51.35	71.72
	ConVEx	71.5	77.6	79.0	84.5	84.0	73.8	67.4	76.8
	Ma2021	89.39	75.11	77.18	84.16	73.53	82.29	72.51	79.17
	L-TapNet+CDT	71.64	67.16	75.88	84.38	82.58	70.05	73.41	75.01
	ESD	84.28	64.36	80.42	83.59	82.64	80.78	81.38	79.63
	JST	88.61	74.54	86.16	89.34	89.19	86.77	83.69	85.47
	JST-GPT	89.31	74.92	86.36	89.57	89.82	87.93	84.86	**86.11**

Table 3
Ablation study: F1 scores on Few-NERD(INTER) 5-way 5-shot are reported.

Ablation models	1shot	5shot
JST-GPT	**75.37**	**79.86**
w/o GPT	70.52	78.94
w/o OA and GPT	68.43	76.43
Only span module	66.84	75.42
Only token module	61.47	66.93
Only hybrid module	62.69	67.63

Table 4
F1 scores under Few-NERD(INTER) 5-way 5-shot setting.

	w OA	w/o OA
Span module	75.42	74.35
Hybrid module	67.63	66.04

that OA plays a crucial role in enhancing the model's performance by creating a distinct separation between the other class and the entities.

4.5.4. Impact of GPT data augmentation

GPT data augmentation significantly improves our model's performance. By increasing the number of training samples, the model can learn a broader range of feature representations and contextual information, improving its ability to recognize variations and different contexts. Data augmentation aids in capturing diverse patterns and reducing the risk of overfitting to the limited available samples. As shown in Table 1, in the 1shot setting, where each entity serves as a prototype, data augmentation increases reliability and sample diversity, resulting in better classification results. However, the impact is less significant in the 5shot setting, where multiple inputs and diverse prototypes are already present. Overall, the integration of GPT data augmentation enhances our model's accuracy and reliability.

5. Conclusion

In this paper, we propose a Joint Span and Toke framework for solving Few-shot NER. To overcome the problems of previous span-based approaches, we use the span feature to extract the boundary features of the entity and the token feature to extract the semantic features of each token that constitutes the entity. For the negative impacts caused by other class, we propose Other Class Attention, which distance entities from the other class in semantic space, and make it easier for the model to distinguish between entities and other class. We use GPT for data augmentation, and the generated data can improve sample diversity and model reliability. Extensive experiments on popular Few-shot NER datasets Few-NERD and SNIPS show that our model outperforms the state-of-the-art model, and our model is more reliable and transferable than the previous model.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The State Key Program of National Natural Science of China, Grant/Award Number: 61533018; National Natural Science Foundation of China, Grant/Award Number: 61402220; The Philosophy and Social Science Foundation of Hunan Province, Grant/Award Number: 16YBA323; Natural Science Foundation of Hunan Province, Grant/Award Number: 2020JJ4525,2022JJ30495; Scientific Research Fund of Hunan Provincial Education Department, Grant/Award Number: 18B279, 19A439, 22A0316.

Appendix A. Supplementary material

https://github.com/fsfwl11406/fewnerd-gptdata.

References

Athiwaratkun, B., Santos, C.N.d., Krone, J., Xiang, B., 2020. Augmented natural language for generative sequence labeling. arXiv preprint arXiv:2009.13272.
Chiu, J.P., Nichols, E., 2016. Named entity recognition with bidirectional LSTM-CNNs. Trans. Assoc. Comput. Linguist. 4, 357–370.
Cui, L., Wu, Y., Liu, J., Yang, S., Zhang, Y., 2021. Template-based named entity recognition using BART. arXiv preprint arXiv:2106.01760.
Das, S.S.S., Katiyar, A., Passonneau, R.J., Zhang, R., 2021. Container: Few-shot named entity recognition via contrastive learning. arXiv preprint arXiv:2109.07589.
Devlin, J., Chang, M.-W., Lee, K., Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.
Ding, N., Xu, G., Chen, Y., Wang, X., Han, X., Xie, P., Zheng, H.-T., Liu, Z., 2021. Few-nerd: A few-shot named entity recognition dataset. arXiv preprint arXiv:2105.07464.
Elsken, T., Staffler, B., Metzen, J.H., Hutter, F., 2020. Meta-learning of neural architectures for few-shot learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12365–12375.
Finn, C., Abbeel, P., Levine, S., 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In: International Conference on Machine Learning. PMLR, pp. 1126–1135.
Fritzler, A., Logacheva, V., Kretov, M., 2019. Few-shot classification in named entity recognition task. In: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing. pp. 993–1000.

Gao, T., Han, X., Liu, Z., Sun, M., 2019. Hybrid attention-based prototypical networks for noisy few-shot relation classification. In: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33. pp. 6407–6414.

Gong, Y., Mao, L., Li, C., 2021. Few-shot learning for named entity recognition based on BERT and two-level model fusion. Data Intell. 3 (4), 568–577.

Henderson, M., Vulić, I., 2020. ConVEx: Data-efficient and few-shot slot labeling. arXiv preprint arXiv:2010.11791.

Hochreiter, S., Younger, A.S., Conwell, P.R., 2001. Learning to learn using gradient descent. In: International Conference on Artificial Neural Networks. Springer, pp. 87–94.

Hou, Y., Che, W., Lai, Y., Zhou, Z., Liu, Y., Liu, H., Liu, T., 2020. Few-shot slot tagging with collapsed dependency transfer and label-enhanced task-adaptive projection network. arXiv preprint arXiv:2006.05702.

Kingma, D.P., Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C., 2016. Neural architectures for named entity recognition. arXiv preprint arXiv:1603.01360.

Li, Z., Gan, Z., Zhang, B., Chen, Y., Wan, J., Liu, K., Zhao, J., Liu, S., 2021a. Semi-supervised noisy label learning for Chinese clinical named entity recognition. Data Intell. 3 (3), 389–401.

Li, X., Wen, Q., Lin, H., Jiao, Z., Zhang, J., 2021b. Overview of CCKS 2020 Task 3: named entity recognition and event extraction in Chinese electronic medical records. Data Intell. 3 (3), 376–388.

Lin, Q., Liu, Y., Wen, W., Tao, Z., Ouyang, C., Wan, Y., 2022. Ensemble making few-shot learning stronger. Data Intell. 4 (3), 529–551.

Ma, X., Hovy, E., 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. arXiv preprint arXiv:1603.01354.

Ma, T., Jiang, H., Wu, Q., Zhao, T., Lin, C.-Y., 2022. Decomposed meta-learning for few-shot named entity recognition. arXiv preprint arXiv:2204.05751.

Ma, J., Yan, Z., Li, C., Zhang, Y., 2021. Frustratingly simple few-shot slot tagging. In: Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021. pp. 1028–1033.

Munkhdalai, T., Yu, H., 2017. Meta networks. In: International Conference on Machine Learning. PMLR, pp. 2554–2563.

Ravi, S., Larochelle, H., 2016. Optimization as a model for few-shot learning.

Snell, J., Swersky, K., Zemel, R., 2017. Prototypical networks for few-shot learning. Adv. Neural Inf. Process. Syst. 30.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. 15 (1), 1929–1958.

Van der Maaten, L., Hinton, G., 2008. Visualizing data using t-SNE. J. Mach. Learn. Res. 9 (11).

Vanschoren, J., 2018. Meta-learning: A survey. arXiv preprint arXiv:1810.03548.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al., 2016. Matching networks for one shot learning. Adv. Neural Inf. Process. Syst. 29.

Wang, P., Xu, R., Liu, T., Zhou, Q., Cao, Y., Chang, B., Sui, Z., 2021. An enhanced span-based decomposition method for few-shot sequence labeling. arXiv preprint arXiv:2109.13023.

Wen, C., Chen, T., Jia, X., Zhu, J., 2021. Medical named entity recognition from un-labelled medical records based on pre-trained language models and domain dictionary. Data Intell. 3 (3), 402–417.

Yang, Y., Katiyar, A., 2020. Simple and effective few-shot named entity recognition with structured nearest neighbor learning. arXiv preprint arXiv:2010.02405.

Yu, D., He, L., Zhang, Y., Du, X., Pasupat, P., Li, Q., 2021. Few-shot intent classification and slot filling with retrieved examples. arXiv preprint arXiv:2104.05763.