An Enhanced Span-based Decomposition Method for Few-Shot Sequence Labeling

Peiyi Wang[1*‡], **Runxin Xu**[1*], **Tianyu Liu**[2], **Qingyu Zhou**[2],
Yunbo Cao[2], **Baobao Chang**[1], **Zhifang Sui**[1†]
[1] Key Laboratory of Computational Linguistics, Peking University, MOE, China
[2] Tencent Cloud Xiaowei
{wangpeiyi9979, runxinxu}@gmail.com
{rogertyliu, qingyuzhou, yunbocao}@tencent.com
{chbb, szf}@pku.edu.cn

Abstract

Few-Shot Sequence Labeling (FSSL) is a canonical paradigm for the tagging models, e.g., named entity recognition and slot filling, to generalize on an emerging, resource-scarce domain. Recently, the metric-based meta-learning framework has been recognized as a promising approach for FSSL. However, most prior works assign a label to each token based on the token-level similarities, which ignores the integrality of named entities or slots. To this end, in this paper, we propose **ESD**, an **E**nhanced **S**pan-based **D**ecomposition method for FSSL. ESD formulates FSSL as a span-level matching problem between test query and supporting instances. Specifically, ESD decomposes the span matching problem into a series of span-level procedures, mainly including enhanced span representation, class prototype aggregation and span conflicts resolution. Extensive experiments show that ESD achieves the new state-of-the-art results on two popular FSSL benchmarks, FewNERD and SNIPS, and is proven to be more robust in the nested and noisy tagging scenarios. Our code is available at https://github.com/Wangpeiyi9979/ESD.

1 Introduction

Many natural language understanding tasks can be formulated as sequence labeling tasks, such as *Named Entity Recognition* (NER) and *Slot Filling* (SF) tasks. Most prior works on sequence labeling follow the supervised learning paradigm, which requires large-scale annotated data and is limited to pre-defined classes. In order to generalize on the emerging, resource-scare domains, Few-Shot Sequence Labeling (FSSL) has been proposed (Hou et al., 2020; Yang and Katiyar, 2020). In FSSL, the model (typically trained on the source domain data) needs to solve the N-way (N unseen classes) K-shot (only K annotated examples for each class) task in the target domain. Figure 1 shows a 2-way 2-shot target domain few-shot NER task, where *'PER'* and *'ORG'* are 2 unseen entity types, and in the training set $\mathcal{S} = \{S_1, S_2\}$, both of them have only 2 annotated entities. The tagging models should annotate *'Albert Einstein'* in the test sentence q as a *'PER'* according to $\mathcal{S}$.

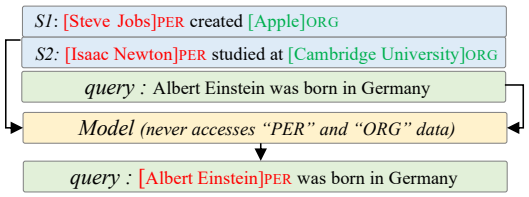

Figure 1: A 2-way 2-shot few-shot NER task. The model needs to learn new entities with few examples.

Recently, *metric-based meta-learning (MBML) methods* have become the mainstream and state-of-the-art methods in FSSL (Hou et al., 2020; Ding et al., 2021), which train the models on the tasks sampled from the source domain sentences in order to mimic and solve the target task. In each task of training and testing, they make the prediction through modeling the similarity between the training set (support set) and the test sentence (query). Specifically, previous MBML methods (Ding et al., 2021; Hou et al., 2020; Yang and Katiyar, 2020) mainly formulate FSSL as a token-level matching problem, which assigns a label to each token based on the token-level similarities. For example, the token *'Albert'* would be labeled as *'B-PER'* due to its resemblance to *'Steve'* and *'Isaac'*.

However, for the MBML methods, selecting the appropriate metric granularity would be fundamental to the success. We argue that prior works that focus on modeling the token-level similarity are sub-optimal in FSSL: 1) they ignore the integrality of named entities or dialog slots that are composed of a text span instead of a single token. 2) in FSSL,

*Equal contribution.
†Corresponding author.
‡Contribution during internship in Tencent.

Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 5012 - 5024

the conditional random fields (CRF) is an important component for the token-level sequence labeling models (Yang and Katiyar, 2020; Hou et al., 2020), but the transition probability between classes in the target domain task can not be sufficiently learned with very few examples (Yu et al., 2021). The previous methods resort to estimate the values with the abundant source domain data, which may suffer from domain shift problem (Hou et al., 2020). To overcome these drawbacks of the token-level models, in this paper, we propose ESD, an **E**nhanced **S**pan-based **D**ecomposition model that formulates FSSL as a span-level matching problem between test query and support set instances.

Specifically, ESD decomposes the span matching problem into three main subsequent span-level procedures. 1) **Span representation**. We find the span representation can be enhanced by information from other spans in the same sentence and the interactions between query and support set. Thus we propose a span-enhancing module to reinforce the span representation by intra-span and inter-span interactions. 2) **Span prototype aggregation**. MBML methods usually aggregate the span vectors that belongs to the same class in the support set to form the class prototype representation. Among all class prototypes, the O-type serves as negative examples and covers all the miscellaneous spans that are not entities, which poses new challenges to identify the entity boundaries. To this end, we propose a span-prototypical module to divide O-type spans into 3 sub-types to distinguish the miscellaneous semantics by their relative position with recognized entities, together with a dynamically aggregation mechanism to form the specific prototype representation for each query span. 3) **Span conflicts resolution**. In the span-level matching paradigm, the predicted spans may conflict with each other. For example, a model may annotate both "Albert Einstein" and "Albert" as *"PER"*. Therefore, we propose a span refining module that incorporates the Soft-NMS (Bodla et al., 2017; Shen et al., 2021) algorithm into the beam search to alleviate this conflict problem. With Beam Soft-NMS, ESD can also handle nested tagging cases without any extra training, which previous methods are incapable of.

We summarize our contribution as follows: (1) We propose ESD, an enhanced span-based decomposition model, which formulates FSSL as a span-level matching problem. (2) We decompose the span matching problem into 3 main subsequent procedures, which firstly produce enhanced span representation, then distinguish miscellaneous semantics of O-types, and achieve the specific prototype representation for each query, and finally resolve span conflicts . (3) Extensive experiments show that ESD achieves new state-of-the-art performance on both few-shot NER and slot filling benchmarks and that ESD is more robust than other methods in nested and noisy scenarios.

2 Related Work

2.1 Few-Shot Learning

Few-Shot Learning (FSL) aims to enable the model to solve the target domain task with a very small training set D_{train} (Wang et al., 2020). In FSL, people usually consider the N-way K-shot task, where the D_{train} has N classes, and each class has only K annotated examples (K is very small, e.g., 5). Training a model only based on D_{train} from scratch will inevitably lead to over-fitting. Therefore, researchers usually introduce the source domain annotated data to help train models, e.g., Few-Shot Relation Classification (FSRC) (Han et al., 2018), Few-Shot Text Classification (FSTC) (Geng et al., 2019) and Few-Shot Event Classification (FSEC) (Wang et al., 2021a). The source domain data do not contain examples belonging to classes in D_{train}, and thus the FSL setting can be guaranteed. Specifically, FSSL tasks in our paper also have the source domain data.

2.2 Metric-Based Meta-Learning

Meta-learning (Hochreiter, Younger, and Conwell, 2001a) is a popular method to deal with Few-Shot Learning (FSL). Meta-learning constructs a series of tasks sampled from the source domain data to mimic the target domain task, and trains models across these sampled tasks. Each task contains a training set (support set) and a test instance (query). The core idea of meta-learning is to help model learn the ability to quickly adapt to new tasks, i.e., learn to learn (Hochreiter, Younger, and Conwell, 2001b). Meta-learning can be combined with the metric-learning (Kulis et al., 2013) (metric-based meta-learning), which makes predictions based on the similarity of the support set and the query. For example, Prototypical Network (Snell, Swersky, and Zemel, 2017) first learns prototype vectors from a few examples in the support set for classes, and further uses prototype vectors for query prediction. Specifically, ESD (our model) follows this

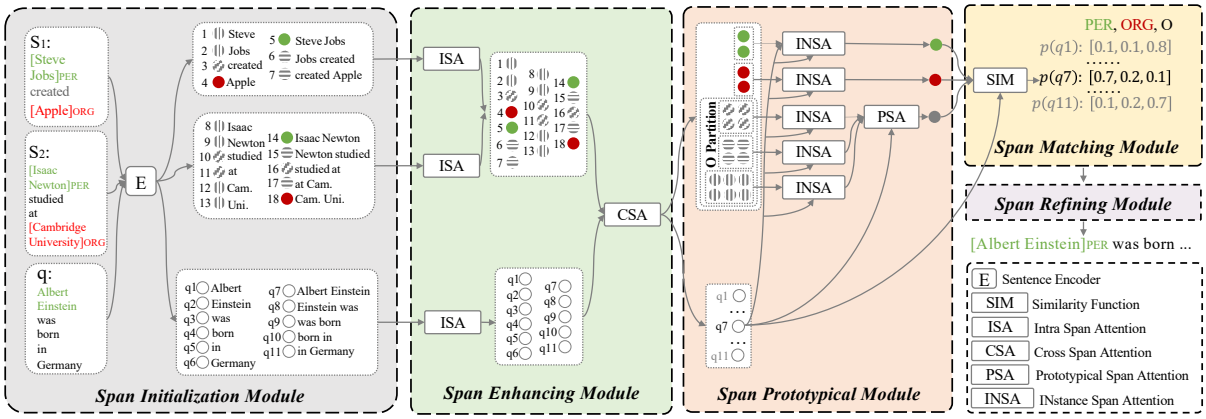

Figure 2: The architecture (5 span modules) of ESD with a 2-way ('PER' and 'ORG') 2-shot input task. We only list spans with lengths less than 2 for clarity. ESD assigns label 'PER' to the span "Albert Einstein" and 'O' to the other spans in query q based on the support set $\{S_1, S_2\}$.

metric-based meta-learning paradigm.

2.3 Few-Shot Sequence Labeling

Recently, few shot sequence labeling (FSSL) has been widely explored by researchers. For example, (Fritzler, Logacheva, and Kretov, 2019a) leverages the prototypical network (Snell, Swersky, and Zemel, 2017) in the few-shot NER. (Yang and Katiyar, 2020) further proposes a cheap but effective method to capture the label dependencies between entity tags without expensive CRF training. (Wang et al., 2021b) utilizes a large unlabelled dataset and proposes a distillation method. (Cui et al., 2021) introduces a prompt method to tap the potential of BART (Lewis et al., 2020). (Hou et al., 2020) extends the TapNet (Yoon, Seo, and Moon, 2019) and proposes a collapsed CRF for few-shot slot filling. (Ma et al., 2021) and (Athiwaratkun et al., 2020) formulate FSSL as a machine comprehension problem and a generation problem, respectively. (Yu et al., 2021) proposed to retrieve the most similar exemplars in the support set for span-level prediction. Although their methods also involve span matching, their main focus is on the retrieval-augmented training. Our work differs from (Yu et al., 2021) in that we propose an effective actionable pipeline to get enhanced span representation, handle miscellaneous semantics of O-types and resolve the potential span conflicts in both non-nested and nested situations, where the last two modules are essential to align the candidate spans with class prototypes in the support set but missing in (Yu et al., 2021). Besides, our enhanced span representation greatly improves the batched softmax objective of (Yu et al., 2021) by inter- and intra-span interactions.

3 Task Formulation

We define a sentence as $\mathbf{x}$ and its label as $\mathbf{y} = \{(s_i, y_i)\}_{i=1}^{M}$, where s_i is a span of $\mathbf{x}$ (e.g., *'Steve Jobs'*), y_i is the label of s_i (e.g., *'PER'*) and M is the number of spans in the $\mathbf{x}$. Following the previous FSSL setting (Hou et al., 2020; Ding et al., 2021), we have data in source domain $\mathcal{D}_{source}$ and target domain $\mathcal{D}_{target}$, and models are evaluated on tasks from $\mathcal{D}_{target}$. Meta-learning based FSSL has two stages, meta-training and meta-testing. In the meta-training stage, the model is trained on tasks sampled from $\mathcal{D}_{source}$ to mimic the test situation, and in the meta-testing stage, the model is evaluated on test tasks. A task is defined as $\mathcal{T} = \{\mathcal{S}, q\}$, consisting of a support set $\mathcal{S} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{I}$, and a query sentence $q = \mathbf{x}$. $\mathcal{S}$ includes N types of entities or slots (N-way), and each type has K annotated examples (K-shot). For spans that do not belong to the N types, e.g., *'studied at'* in Figure 1, we set their label to O. The types except O in each test task are guaranteed to not exist in the training tasks. Given a task $\mathcal{T} = \{\mathcal{S}, q\}$, the model needs to assign a label to each span in the query sentence q based on $\mathcal{S}$.

4 Methodology

Our ESD formulates FSSL as a span-level matching problem and decomposes it into a series of span-related procedures for a better span matching. Figure 2 illustrates the architecture of ESD.

4.1 Span Initialization Module

Given a task $\mathcal{T} = \{\mathcal{S}, q\}$, we use BERT (Devlin et al., 2019) to encode sentences in $\mathcal{S}$ and q, and utilize the output of the last layer to represent tokens in the sentence. Therefore, for a sentence with N tokens $\mathbf{x} = (x_1, x_2, ..., x_N)$, we can achieve representations $(\mathbf{h_1}, \mathbf{h_2}, ..., \mathbf{h_N})$, where $\mathbf{h_i} \in \mathbb{R}^{d_w}$ is the hidden state corresponding to the token x_i. Then, for a span $s = (l, r)$, where l and r are the start index and end index of span s in the sentence $\mathbf{x}$, we obtain its initial representation $\mathbf{s}_{(l,r)} = [\mathbf{h_l}; \mathbf{h_r}]\mathbf{W_s}$.[1] We enumerate spans in the sentence with a maximum length of L.

4.2 Span Enhancing Module

4.2.1 Intra Span Attention

Intuitively, the meaning of specific spans can be inferred from other spans in the same sentence. We thus design an Intra Span Attention (ISA) mechanism. Given all the span representations of a sentence $\mathbf{S} \in \mathbb{R}^{B \times d}$ (B is the number of spans). We denote the i-th row of $\mathbf{S}$ as $\mathbf{s_i}$, which represents the i-th span in the sentence. For $\mathbf{s_i}$, we first get its ISA span representation $\bar{\mathbf{s}}_\mathbf{i} = \sum_{j=1}^{B} \alpha_\mathbf{i}{}^j \mathbf{s_j}$, where $\alpha_\mathbf{i} = \text{softmax}(\mathbf{s_i}\mathbf{S^T})$. For clarity, we denote this attention aggregation operation as ϕ, i.e.,

$$\bar{\mathbf{s}}_\mathbf{i} = \phi(\mathbf{s_i}, \mathbf{S}) \tag{1}$$

then, a Feed Forward Neural Networks (FFN) (Vaswani et al., 2017) with residual connection (He et al., 2016) and layer normalization (Ba, Kiros, and Hinton, 2016) are used to get the final ISA enhanced feature $\tilde{\mathbf{s}}_\mathbf{i}$,

$$\tilde{\mathbf{s}}_\mathbf{i} = \text{LayerNorm}(\mathbf{s_i} + \text{FFN}_{\text{isa}}(\bar{\mathbf{s}}_\mathbf{i})) \tag{2}$$

$$\text{FFN}_{\text{isa}}(\bar{\mathbf{s}}_\mathbf{i}) = \text{GELU}(\bar{\mathbf{s}}_\mathbf{i}\mathbf{W^1_{isa}})\mathbf{W^2_{isa}} \tag{3}$$

4.2.2 Cross Span Attention

After ISA, to accommodate the span matching between the test query and supporting sentences, and facilitate the inter-span interaction, we propose Cross Span Attention (CSA) to enhance query spans $\tilde{\mathbf{Q}} \in \mathbb{R}^{B_q \times d}$ with the support set spans $\{\tilde{\mathbf{S}}_\mathbf{i} \in \mathbb{R}^{B_i \times d}; i = 1, ..., I\}$, and vice versa. We first concatenate all span representations in the support set into one matrix $\tilde{\mathbf{S}} = [\tilde{\mathbf{S}}_\mathbf{1}, \tilde{\mathbf{S}}_\mathbf{2}, ..., \tilde{\mathbf{S}}_\mathbf{I}] \in \mathbb{R}^{B_s \times d}$. We denote the n-th row of $\tilde{\mathbf{S}}$ as $\tilde{\mathbf{s}}_\mathbf{n}$ and the m-th row of $\tilde{\mathbf{Q}}$ as $\tilde{\mathbf{q}}_\mathbf{m}$, and obtain their CSA span representations $\hat{\mathbf{s}}_\mathbf{n} = \phi(\tilde{\mathbf{s}}_\mathbf{n}, \tilde{\mathbf{Q}})$ and $\hat{\mathbf{q}}_\mathbf{m} = \phi(\tilde{\mathbf{q}}_\mathbf{m}, \tilde{\mathbf{S}})$. Then, we get the final CSA enhanced representation $\check{\mathbf{s}}_\mathbf{n}$ and $\check{\mathbf{q}}_\mathbf{m}$ as follows,

$$\check{\mathbf{s}}_\mathbf{n} = \text{LayerNorm}(\tilde{\mathbf{s}}_\mathbf{n} + \text{FFN}_{\text{csa}}(\hat{\mathbf{s}}_\mathbf{n})) \tag{4}$$

$$\check{\mathbf{q}}_\mathbf{m} = \text{LayerNorm}(\tilde{\mathbf{q}}_\mathbf{m} + \text{FFN}_{\text{csa}}(\hat{\mathbf{q}}_\mathbf{m})) \tag{5}$$

$$\text{FFN}_{\text{csa}}(\mathbf{x}) = \text{GELU}(\mathbf{x}\mathbf{W^1_{csa}})\mathbf{W^2_{csa}} \tag{6}$$

4.3 Span Prototypical Module

4.3.1 Instance Span Attention

Since different support set spans play different roles for a query span, we propose multi-INstance Span Attention (INSA) to get class representations. For the i-th class that contains K annotated spans with enhanced representations $\check{\mathbf{S}}_\mathbf{i} = [\check{\mathbf{s}}_\mathbf{i}^1, ..., \check{\mathbf{s}}_\mathbf{i}^K]$ in the support set, given a query span $\check{\mathbf{q}}_\mathbf{m}$, INSA gets the corresponding prototypical representation $\mathbf{z}_\mathbf{m}^\mathbf{i} = \phi(\check{\mathbf{q}}_\mathbf{m}, \check{\mathbf{S}}_\mathbf{i})$.

4.3.2 O Partition and Prototypical Span Attention

The O-type spans have huge quantities and miscellaneous semantics, which is hard to be represented by only one prototypical vector. In the span-based framework, considering the boundary information is essential for a span, we divide the O-type spans into 3 sub-classes according to their boundary, to alleviate their miscellaneous semantics problem. Specifically, given a sentence with I annotated spans $\{(l_i, r_i)\}_{i=1}^{I}$, where l_i and r_i are the left and right boundary of the i-th annotated span. For each of the other spans (l_o, r_o) , we assign it a sub-class O_{sub} as follows,

$$O_{sub} = \begin{cases} O_1, & \forall i, s.t.\ r_o < l_i \vee l_o > r_i \\ O_2, & \exists i, s.t.\ l_o \geq l_i \wedge r_o \leq r_i \\ O_3, & \text{Others} \end{cases} \tag{7}$$

where O_1 denotes the span that does not overlap with any entities or slots in the sentence, e.g., *"study at"* in S_2 of Figure 2, and O_2 represents the span that is the sub-span of an entity or slot, e.g., *"Isaac"* in S_2 of Figure 2. After O Partition, we get the prototypical representation of each O_{sub} via INSA, thus for a query span $\check{\mathbf{q}}_\mathbf{m}$, we have 3 sub-class representations $\mathbf{Z}_\mathbf{m}^\mathbf{o} = [\mathbf{z}_\mathbf{m}^{\mathbf{o_1}}, \mathbf{z}_\mathbf{m}^{\mathbf{o_2}}, \mathbf{z}_\mathbf{m}^{\mathbf{o_3}}]$ for the class O. Then, we utilize Prototypical Span Attention (PSA) to achieve the final O representation $\mathbf{z}_\mathbf{m}^\mathbf{o} = \phi(\check{\mathbf{q}}_\mathbf{m}, \mathbf{Z}_\mathbf{m}^\mathbf{o})$.

4.4 Span Matching Module

Given a task $\mathcal{T} = (\mathcal{S}, q)$, for the m-th span in q_m, we achieve its enhanced representation $\check{\mathbf{q}}_\mathbf{m}$

[1] We omit the bias term in this paper for clarity.

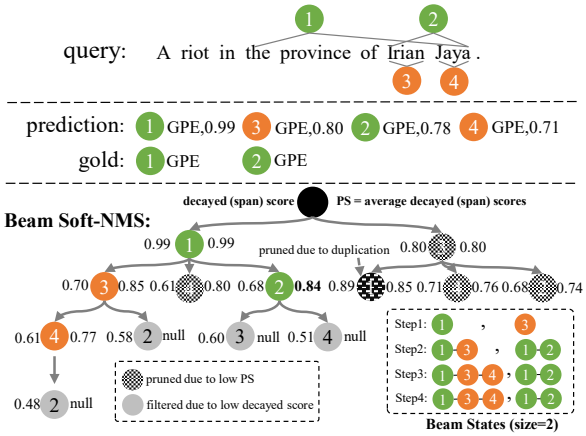

Figure 3: A demonstration for Beam Soft-NMS (BSNMS). For clarity, we set the filter threshold δ to 0.6, and suppose the span score is always decayed by the overlapped spans with a constant decayed score -0.1 (Note that the example is simplified for demonstration). BSNMS filters false positive spans (colored in orange) in this demonstration. A more clear step by step demonstration is included in Appendix D.

and corresponding prototypical vectors $\mathbf{Z_m} = (\mathbf{z_m^o}, \mathbf{z_m^1}, ..., \mathbf{z_m^N})$ through previous span modules. Then, we predict q_m as the type z_k in the support set with probability,

$$p(y_m = z_k|q_m) = \frac{exp(-L_2(\check{\mathbf{q}}_\mathbf{m}, \mathbf{z_k^m}))}{\sum_{k'} exp(-L_2(\check{\mathbf{q}}_\mathbf{m}, \mathbf{z_{k'}^m}))} \tag{8}$$

where L_2 is the euclidean distance. During training, we use cross-entropy as our loss function,

$$\mathcal{L} = -\frac{1}{B_q}\sum_{m=1}^{B_q} log\ p(y_m^*|q_m) \tag{9}$$

where y_m^* is the gold label of q_m and B_q is the number of spans in the query q.

4.5 Span Refining Module in Inference

During inference, spans outputted by the matching module may have conflicts, we thus propose a refining module that incorporates the SoftNMS into beam search for the conflicts resolution. For the m-th span with the left index l_m and the right index r_m in the query, we obtain its prediction probability distribution $\mathbf{p_m}$, label $y_m = \text{argmax}(\mathbf{p_m})$ and score $score_m = \max(\mathbf{p_m})$. Figure 3 illustrates a simplified post-processing process. In each step, we first expand all beam states (e.g., states 1-3 and 1-2 in step2 of Figure 3), and then prune new states according to the beam size. Specifically, given a beam state S containing spans $\{l_t, r_t, score_t, y_t\}_{t=1}^T$, for each non-contained span $s_i = (l_i, r_i, score_i, y_i)$, we first calculate its decayed score $score_i^{decay}$,

$$score_i^{decay} = score_i * u^\eta \tag{10}$$

$$\eta = \sum_{t=1}^{T} \mathbb{I}(\text{IoU}(s_i, s_t) \geq k) \tag{11}$$

where $\text{IoU}(s_i, s_t) = \frac{|\{l_i,...,r_i\} \cap \{l_t,...,r_t\}|}{|\{l_i,...,r_i\} \cup \{l_t,...,r_t\}|}$ is the overlap ratio of two spans. The decay ratio u and threshold k are hyperparameters. Then, we expand the beam state S with the non-contained span s_i if $score_i^{decay} > \delta$. For example, in the step2 of Figure 3, we expand the state 1-3 to 1-3-4, while the state 1-3-2 fails to be expanded since $score_2^{decay} = 0.58 <= \delta$. After expanding all states, we prune available beam states with lower path scores. For example, we prune states 1-4, 3-4 and 3-2 in step2 of Figure 3. In addition, as our needed output is order-independent, we also prune duplicate states, e.g., the state 3-1 (equivalent to the state 1-3) for the diversity of beam states. When all states in the beam can not be expanded or have been expanded before but failed, we select the beam with the largest path score as the final output.

5 Experiments

5.1 Experiments Setup

Datasets We evaluate ESD on FewNERD (Ding et al., 2021) and SNIPS (Coucke et al., 2018). **FewNERD** designs an annotation schema of 8 coarse-grained (e.g., 'Person') entity types and 66 fine-grained (e.g., 'Person-Artist') entity types, and constructs two tasks. One is **FewNERD-INTRA**, where all the entities in the training set (source domain), validation set and test set (target domain) belong to different coarse-grained types. The other is **FewNERD-INTER**, where only the fine-grained entity types are mutually disjoint in different sets. For the sake of sampling diversity, FewNERD adopts the N-way $K \sim 2K$-shot sampling method to construct tasks (each class in the support set has $K \sim 2K$ annotated entities). Both FewNERD-INTRA and FewNERD-INTER have 4 settings, 5-way $1 \sim 2$-shot, 5-way $5 \sim 10$-shot, 10-way $1 \sim 2$-shot and 10-way $5 \sim 10$-shot. **SNIPS** is a slot filling dataset, which contains 7 domains $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_7\}$. (Hou et al., 2020) constructs few-shot slot filling task with the leave-one-out strategy, which means when testing on the target domain $\mathcal{D}_i$, they randomly chose $\mathcal{D}_j (i \neq j)$

Models	FEW-NERD (INTRA)					FEW-NERD (INTER)				
	1 ∼ 2 shot		5 ∼ 10 shot		Avg.	1 ∼ 2 shot		5 ∼ 10 shot		Avg.
	5 way	10 way	5 way	10 way		5 way	10 way	5 way	10 way	
ProtoBERT	20.76±0.84	15.04±0.44	42.54±0.94	35.40±0.13	28.44	38.83±1.49	32.45±0.79	58.79±0.44	52.92±0.37	45.75
NNShot	25.78±0.91	18.27±0.41	36.18±0.79	27.38±0.53	26.90	47.24±1.00	38.87±0.21	55.64±0.63	49.57±2.73	47.83
StructShot	30.21±0.90	21.03±1.13	38.00±1.29	26.42±0.60	28.92	51.88±0.69	43.34±0.10	57.32±0.63	49.57±3.08	50.53
ESD (Ours)	**36.08**±1.6	**30.00**±0.70	**52.14**±1.5	**42.15**±2.6	**40.09**	**59.29**±1.25	**52.16**±0.79	**69.06**±0.80	**64.00**±0.43	**61.13**

Table 1: F1 scores with standard deviations on FewNERD. The best results are in **boldface**.

	Models	We	Mu	Pl	Bo	Se	Re	Cr	Avg.
1-SHOT	TransferBERT	55.82±2.75	38.01±1.74	45.65±2.02	31.63±5.32	21.96±3.98	41.79±3.81	38.53±7.42	39.06±3.86
	MN+BERT	21.74±4.60	10.68±1.07	39.71±1.81	58.15±0.68	24.21±1.20	32.88±0.64	69.66±1.68	36.72±1.67
	ProtoBERT	46.72±1.03	40.07±0.48	50.78±2.09	68.73±1.87	60.81±1.70	55.58±3.56	67.67±1.16	55.77±1.70
	Ma2021	-	-	-	-	-	-	-	69.3(unk)
	L-TapNet+CDT	71.53±4.04	60.56±0.77	66.27±2.71	84.54±1.08	76.27±1.72	70.79±1.60	62.89±1.88	70.41±1.97
	ESD (Ours)	78.25±1.50	54.74±1.02	71.15±1.55	71.45±1.38	67.85±0.75	71.52±0.98	78.14±1.46	**70.44**±0.47
5-SHOT	TransferBERT	59.41±0.30	42.00±2.83	46.07±4.32	20.74±3.36	28.20±0.29	67.75±1.28	58.61±3.67	46.11±2.29
	MN+BERT	36.67±3.64	33.67±6.12	52.60±2.84	69.09±2.36	38.42±4.06	33.28±2.99	72.10±1.48	47.98±3.36
	ProtoBERT	67.82±4.11	55.99±2.24	46.02±3.19	72.17±1.75	73.59±1.60	60.18±6.96	66.89±2.88	63.24±3.25
	Retriever	82.95(unk)	61.74(unk)	71.75(unk)	81.65(unk)	73.10(unk)	79.54(unk)	51.35(unk)	71.72(unk)
	ConVEx	71.5(unk)	77.6(unk)	79.0(unk)	84.5(unk)	84.0(unk)	73.8(unk)	67.4(unk)	76.8(unk)
	Ma2021	89.39(unk)	75.11(unk)	77.18(unk)	84.16(unk)	73.53(unk)	82.29(unk)	72.51(unk)	79.17(unk)
	L-TapNet+CDT	71.64±3.62	67.16±2.97	75.88±1.51	84.38±2.81	82.58±2.12	70.05±1.61	73.41±2.61	75.01±2.46
	ESD (Ours)	84.50±1.06	66.61±2.00	79.69±1.35	82.57±1.37	82.22±0.81	80.44±0.80	81.13±1.84	**79.59**±0.39

Table 2: F1 scores with standard deviations on 7 domains of SNIPS. The best results are in **boldface**. 'unk' denotes methods that do not report deviations in their paper. Baselines of 1-shot and 5-shot settings are different since ConVEx and Retriever do not report the 1-shot results in their paper.

as the validation domain, and train the model on source domains $\mathcal{D} - \{\mathcal{D}_i, \mathcal{D}_j\}$. In the sampling task of SNIPS, all classes have K annotated examples in the support set, but the number of them (N) is not fixed. The few-shot slot filling task in each domain of SNIPS has two settings, 1-shot and 5-shot. FSSL models are trained and evaluated on tasks sampled from the source and target domain, respectively. To ensure the fairness, we use the public sampled data provided by (Ding et al., 2021) for FewNERD[2] and data provided by (Hou et al., 2020) for SNIPS, to train and evaluate our model.

Parameter Settings Following previous methods (Hou et al., 2020; Ding et al., 2021), we use uncased BERT-base as our encoder. We use Adam (Kingma and Ba, 2015) as our optimizer. We set the dropout ratio (Srivastava et al., 2014) to 0.1. The maximum span length L is set to 8. For BSNMS, the beam size b is 5. Because FewNERD and SNIPS do not have nested instances, we set the threshold to filter false positive spans δ to 0.1, the threshold to decay span scores k to $1e$-5 and the decay ratio u to $1e$-5 to force the refining results have no nested spans. More details of our parameter settings are provided in Appendix A.

Evaluation Metrics For FewNERD, following (Ding et al., 2021), we report the micro F1 over all test tasks, and the average result of 5 different runs. For SNIPS, following (Hou et al., 2020), we first calculate micro F1 score for each test episode (an episode contains a number of test tasks), and then report the average F1 score for all test episodes as the final result. We report the average result of 10 different runs the same as (Hou et al., 2020).

Baselines For systematic comparisons, we introduce a variety of baselines, including **ProtoBERT** (Ding et al., 2021; Hou et al., 2020), **NNShot** (Ding et al., 2021), **StructShot** (Ding et al., 2021), **TransferBERT** (Hou et al., 2020), **MN+BERT** (Hou et al., 2020), **L-TapNet+CDT** (Hou et al., 2020), **Retriever** (Yu et al., 2021), **ConVEx** (Henderson

[2]The FewNERD data we used is from https://cloud.tsinghua.edu.cn/f/0e38bd108d7b49808cc4/?dl=1, which corresponds to the results reported in https://arxiv.org/pdf/2105.07464v6.pdf.

Ablation Models	F1
ESD	61.7±1.3
r.m. intra span attention (ISA)	61.0±0.4
r.m. cross span attention (CSA)	59.1±1.1
r.m. instance span attention (INSA)	58.2±1.3
r.m. O partition (OP)	60.6±1.3
r.m. beam soft-nms (BSNMS)	57.3±1.4

Table 3: The effect of our proposed mechanisms on the validation set of SNIPS (1-shot, domain "We"). We report the average result of 3 different runs with standard deviations. *r.m.* denotes *remove*.

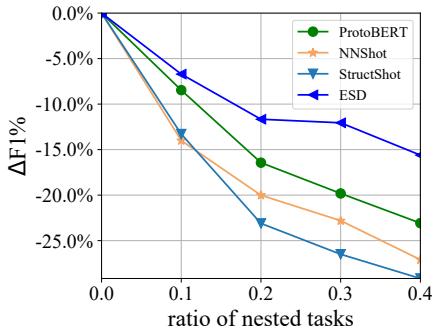

Figure 4: The performance drops in nested situations when increasing r_{nest}. ΔF1% is the F1 reduction percent over $r_{nested} = 0$.

and Vulić, 2021) and **Ma2021** (Ma et al., 2021). Please refer to the Appendix A for more details.

5.2 Main Results

Table 1 illustrates the results on FewNERD. As is shown, among all task settings, ESD consistently outperforms ProtoBERT, NNShot and StructShot by a large margin. For example, compared with StructShot, ESD achieves 11.17 and 10.60 average F1 improvement on INTRA and INTER, respectively. Table 2 shows the results on SNIPS. In the 1-shot setting, L-TapNet+CDT is the previous best method. Compared with L-TapNet+CDT, ESD achieves comparable results and 4.58 F1-scores improvement in the 1-shot and 5-shot settings, respectively. We think the reason is that the information benefits brought by our cross span attention mechanism in 1-shot setting is much less than that in 5-shot setting. In addition, compared with L-TapNet+CDT, ESD performs more stable, and also has a better model efficiency (Please refer to Section 6.3). In 5-shot setting, Ma2021 is previous best method, and ESD outperforms it 1.14 and 0.42 F1-scores in 1-shot and 5-shot settings, respectively. These results prove the effectiveness of ESD in few-shot sequence labeling.

6 Analysis

6.1 Ablation Study

To illustrate the effect of our proposed mechanisms, we conduct ablation studies by removing one component of ESD at a time. Table 3 shows the results on the validation set of SNIPS (1-shot, domain "We"). Firstly, we remove ISA or CSA, which means the span cannot be aware of other spans within the same sentence or spans from other sentences. As shown in Table 3, the average F1 scores drop 0.7 and 2.6 without ISA and CSA, respectively. These results suggest that our span enhancing module with ISA and CSA is effective. In addition, CSA is more effective than ISA, since CSA can enhance query spans with whole support set spans, while ISA enhances spans only with other spans in the same setence. CSA brings much more information than ISA. Secondly, when we remove INSA and achieve the prototypical representation of a class through an average operation, the average F1 score would drop 3.5. When we do not consider the sub-classes of O-type spans (*r.m.* OP), the average F1 score would drop 1.1. These results show that our span prototypical module is necessary. At last, the result without BSNMS suggests the importance of our post-processing algorithm in this span-level few-shot labeling framework. Moreover, with BSNMS, ESD can also easily adapt to the nested situation.

6.2 Robustness in the Nested Situation

Sequence labeling tasks such as NER can have nested situations. For example, in the sentence "Isaac Newton studied at Cambridge University", where "Cambridge" is a location and "Cambridge University" is an organization. However, both FewNERD and SNIPS do not annotate nested examples. To explore the robustness of ESD on such a challenging but common situation, we construct **FewNERD-nested**, which has the same training tasks as FewNERD-INTRA, but different test tasks with a nested ratio r_{nested}. In FewNERD-nested, we sample each test task either from FewNERD or from GENIA (Ohta et al., 2002) with the probability $1 - r_{nested}$ and r_{nested}, respectively, and all tasks sampled from GENIA are guaranteed to have the query with nested entities. We sample validation tasks with nested instances from

Models	F1
ESD (with BSNMS)	31.25
ESD (with SoftNMS)	31.09
ESD (with Beam Search)	28.94
ESD (without Post-processing)	30.34

Table 4: Comparison between different post-processing methods in the nested situation with $r_{nested} = 1$.

Models	#Para.	Inference Time	
		1-SHOT	5-SHOT
ESD	112M	8.53 ms	18.47 ms
ProtoBERT	110M	3.13 ms	5.27 ms
L-TapNet+CDT	110M	24.67 ms	54.19 ms

Table 5: The parameter number and inference time per task of ESD and L-TapNet+CDT in the domain "We" of SNIPS. Although ESD is slower than ProtoBERT, ESD outperforms ProtoBERT by 31.53 and 16.68 F1 scores in 1- and 5-shot settings with a bearable latency.

ACE05 (Walker et al., 2006) to tune k, u and δ in BSNMS. Please refer to the Appendix B for more details about FewNERD-nested. Figure 4 shows the results of ESD and several typical baselines in FewNERD-nested with different r_{nested}. When r_{nested} increases, ESD is more stable than previous methods, since ESD with BSNMS can easily extend to nested tagging cases without any extra training while previous methods are incapable of. In addition, we also compare different post-processing methods when $r_{nested} = 1$. As shown in Table 4, the beam search method can not handle the nested situation, and thus it harms the performance. SoftNMS (Shen et al., 2021) and BSNMS can both improve the model performance. However, when incorporating beam search into SoftNMS, the model can be more flexible and avoid some local optimal post-processing results achieved by SoftNMS (e.g., spans 1,3 and 4 in Figure 3), and thus BSNMS outperforms SoftNMS [3].

6.3 Model Efficiency

Compared with the token-level models, ESD needs to enumerate all spans within the length L for a sentence. Therefore, the number of spans is approximately L times that of tokens, which may bring extra computation overhead. To evaluate the efficiency of ESD, we compare the average inference time per task of ESD (including the BSNMS post-processing process), L-TapNet+CDT (the state-of-the-art token-level baseline model with the open codebase in the SNPIS) and ProtoBERT (an extremely simple token-level baseline). As shown in Table 5, with exactly the same hardware setting, in the domain "We" of SNIPS 1-shot setting, ESD (avg. 8.53 ms per task) is nearly 3 times faster than L-TapNet+CDT (avg. 24.67 ms per task). We see a similar tendency in the 5-shot setting. Although ESD (avg. 8.53 and 18.47 ms in 1- and 5-shot per task) is slower than ProtoBERT (avg. 3.13 and 5.27 ms), it outperforms ProtoBERT by 31.53 and 16.68 F1 scores in 1- and 5-shot settings (reported in Table 2) with a bearable latency. Besides the inference time, we also compare the parameter number of these models. As is shown, the added parameter scale (span enhancing and prototypical modules) is very small (2M) compared with the ProtoBERT and L-TapNet+CDT (110M). These results show that ESD has an acceptable efficiency.

[3] ESD is also more robust than baselines in the noisy situation. Please refer to Appendix C for details.

Models	F1	Total	FP-Span	FP-Type
ESD	59.29	9.4k	72.8%	27.2%
ProtoBERT	38.83	30.4k	86.7%	13.3%
NNShot	47.24	21.7k	84.7%	15.3%
StructShot	51.88	14.5k	80.0%	20.0%

Table 6: Error analysis of 5 way $1 \sim 2$ shot on FewNERD-INTER. 'FP-Span' denotes extracted entities with the wrong span boundary, and 'FP-Type' represents extracted entities with the right span boundary but the wrong entity type. 'Total' denotes the total wrong prediction of two types.

6.4 Error Analysis

To further explore what types of errors the model makes in detail, we divide error of model prediction into 2 categories, 'FP-Span' and 'FP-Type'. As shown in Table 6, ESD outperforms baselines and has much less false positive prediction errors. 'FP-Span' is the major prediction error for all models, showing that it is hard to locate the right span boundary in FSSL for existing models. Therefore, we should pay more attention to the span recognition in the future work. However, compared with previous methods, ESD has less ratio of the 'FP-span' error. We think the reason is that our span-level matching framework with a series of span-related procedures has a better perception of the entity and slot spans than that of our baselines.

7 Conclusion

In this paper, we propose ESD, an enhanced span-based decomposition model for few-shot sequence labeling (FSSL). To overcome the drawbacks of previous token-level methods, ESD formulates FSSL as a span-level matching problem, and decomposes it into a series of span-related procedures, mainly including span representation, class prototype aggregation and span conflicts resolution for a better span matching. Extensive experiments show that ESD achieves the state-of-the-art performance on two popular few-shot sequence labeling benchmarks and that ESD is more robust than previous models in the noisy and nested situation.

Acknowledgements

The authors would like to thank the anonymous reviewers for their thoughtful and constructive comments. This paper is supported by the National Key Research and Development Program of China under Grant No. 2020AAA0106700, the National Science Foundation of China under Grant No.61936012 and 61876004, and NSFC project U19A2065.

References

Athiwaratkun, B.; Nogueira dos Santos, C.; Krone, J.; and Xiang, B. 2020. Augmented Natural Language for Generative Sequence Labeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 375–385. Online: Association for Computational Linguistics.

Ba, J. L.; Kiros, J. R.; and Hinton, G. E. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Bodla, N.; Singh, B.; Chellappa, R.; and Davis, L. S. 2017. Soft-NMS–improving object detection with one line of code. In *Proceedings of the IEEE international conference on computer vision*, 5561–5569.

Coucke, A.; Saade, A.; Ball, A.; Bluche, T.; Caulier, A.; Leroy, D.; Doumouro, C.; Gisselbrecht, T.; Caltagirone, F.; Lavril, T.; et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*.

Cui, L.; Wu, Y.; Liu, J.; Yang, S.; and Zhang, Y. 2021. Template-Based Named Entity Recognition Using BART. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 1835–1845. Online: Association for Computational Linguistics.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics.

Ding, N.; Xu, G.; Chen, Y.; Wang, X.; Han, X.; Xie, P.; Zheng, H.; and Liu, Z. 2021. Few-NERD: A Few-shot Named Entity Recognition Dataset. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 3198–3213. Online: Association for Computational Linguistics.

Fritzler, A.; Logacheva, V.; and Kretov, M. 2019a. Few-shot classification in named entity recognition task. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, 993–1000.

Fritzler, A.; Logacheva, V.; and Kretov, M. 2019b. Few-shot classification in named entity recognition task. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, 993–1000.

Geng, R.; Li, B.; Li, Y.; Zhu, X.; Jian, P.; and Sun, J. 2019. Induction Networks for Few-Shot Text Classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 3904–3913. Hong Kong, China: Association for Computational Linguistics.

Han, X.; Zhu, H.; Yu, P.; Wang, Z.; Yao, Y.; Liu, Z.; and Sun, M. 2018. FewRel: A Large-Scale Supervised Few-Shot Relation Classification Dataset with State-of-the-Art Evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 4803–4809. Brussels, Belgium: Association for Computational Linguistics.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Henderson, M.; and Vulić, I. 2021. ConVEx: Data-Efficient and Few-Shot Slot Labeling. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 3375–3389. Online: Association for Computational Linguistics.

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

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

Hou, Y.; Che, W.; Lai, Y.; Zhou, Z.; Liu, Y.; Liu, H.; and Liu, T. 2020. Few-shot Slot Tagging with Collapsed Dependency Transfer and Label-enhanced Task-adaptive Projection Network. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 1381–1393. Online: Association for Computational Linguistics.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980.

Kulis, B.; et al. 2013. Metric Learning: A Survey. *Foundations and Trends® in Machine Learning*, 5(4): 287–364.

Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; and Zettlemoyer, L. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 7871–7880.

Ma, J.; Yan, Z.; Li, C.; and Zhang, Y. 2021. Frustratingly Simple Few-Shot Slot Tagging. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 1028–1033. Online: Association for Computational Linguistics.

Ohta, T.; Tateisi, Y.; Kim, J.-D.; Mima, H.; and Tsujii, J. 2002. The GENIA corpus: An annotated research abstract corpus in molecular biology domain. In *Proceedings of the human language technology conference*, 73–77. Citeseer.

Shen, Y.; Ma, X.; Tan, Z.; Zhang, S.; Wang, W.; and Lu, W. 2021. Locate and Label: A Two-stage Identifier for Nested Named Entity Recognition. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2782–2794.

Snell, J.; Swersky, K.; and Zemel, R. 2017. Prototypical networks for few-shot learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 4080–4090.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1): 1929–1958.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Vinyals, O.; Blundell, C.; Lillicrap, T.; kavukcuoglu, k.; and Wierstra, D. 2016. Matching Networks for One Shot Learning. In Lee, D.; Sugiyama, M.; Luxburg, U.; Guyon, I.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Walker, C.; Strassel, S.; Medero, J.; and Maeda, K. 2006. ACE 2005 multilingual training corpus. *Linguistic Data Consortium, Philadelphia*, 57: 45.

Wang, P.; Xun, R.; Liu, T.; Dai, D.; Chang, B.; and Sui, Z. 2021a. Behind the Scenes: An Exploration of Trigger Biases Problem in Few-Shot Event Classification. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 1969–1978.

Wang, Y.; Mukherjee, S.; Chu, H.; Tu, Y.; Wu, M.; Gao, J.; and Awadallah, A. H. 2021b. Meta Self-training for Few-shot Neural Sequence Labeling. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 1737–1747.

Wang, Y.; Yao, Q.; Kwok, J. T.; and Ni, L. M. 2020. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys (CSUR)*, 53(3): 1–34.

Yang, Y.; and Katiyar, A. 2020. Simple and Effective Few-Shot Named Entity Recognition with Structured Nearest Neighbor Learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 6365–6375. Online: Association for Computational Linguistics.

Yoon, S. W.; Seo, J.; and Moon, J. 2019. Tapnet: Neural network augmented with task-adaptive projection for few-shot learning. In *International Conference on Machine Learning*, 7115–7123. PMLR.

Yu, D.; He, L.; Zhang, Y.; Du, X.; Pasupat, P.; and Li, Q. 2021. Few-shot Intent Classification and Slot Filling with Retrieved Examples. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 734–749. Online: Association for Computational Linguistics.

A Experiments

A.1 Baselines

We compare ESD with a variety of baselines as follows:

- **TransferBERT** (Hou et al., 2020) is a fine-tuning based model, which is a direct application of BERT (Devlin et al., 2019) to the few-shot sequence labeling.
- **ConVEx** (Henderson and Vulić, 2021) is a fine-tuning based model, which is first pre-trained on the Reddit corpus with the sequence labeling objective tasks and then fine-tuned on the source domain and target domain annotated data for final few shot sequence labelingm
- **Ma2021** (Ma et al., 2021) formulates sequence labeling as the machine reading comprehension problem, and proposes some questions to extract slots in the query sentence.
- **ProtoBERT** (Fritzler, Logacheva, and Kretov, 2019b) predicts the query labels according to the similarity of BERT hidden states of support set and query tokens.
- **Matching Net (MN)+BERT** (Hou et al., 2020) is similar to ProtoBERT. The only difference is that MN uses the matching network (Vinyals et al., 2016) for token classification.
- **L-TapNet-CDT** (Hou et al., 2020) utilizes the task-adaptive projection network (Yoon, Seo, and Moon, 2019), pair-wise embedding and collapsed dependency transfer mechanisms to do classification.
- **NNShot** (Yang and Katiyar, 2020) is similar to ProtoBERT, while it makes the prediction based on the nearest neighbor.
- **StructShot** (Yang and Katiyar, 2020) adopts an additional Viterbi decoder during the inference phase on top of NNShot.
- **Retriever** (Yu et al., 2021) is a retrieval based method which does classification according to the most similar example in the support set.

A.2 Parameter Setting

In our implementation, we utilize *BERT-base-uncased* as our backbone encoder the same as (Hou et al., 2020; Yu et al., 2021; Ding et al., 2021). We use Adam (Kingma and Ba, 2015) as our optimizer. In FewNERD, the learning rate is $2e-5$ for BERT encoder and $5e-4$ for other modules. In the 1-shot setting of SNIPS, for the domain "Mu", the learning rate is $5e-6$ for BERT encoder and $1e-4$ for other modules. For the domain "Bo", the learning rate is $1e-5$ for BERT encoder and $1e-4$ for other modules. For other settings of SNIPS, the learning rate is $5e-5$ for BERT encoder and $5e-4$ for other modules. We set the dropout ratio (Srivastava et al., 2014) to 0.1. The dimension of span representation d and the maximum span length L is set to 100 and 8, respectively. For BSNMS, the beam size b is 5. Since these is no nested instances in FewNERD and SNIPS, we set the threshold to filter false positive spans δ to 0.1, the threshold to decay span scores k to $1e-5$ and the decay ratio u to $1e-5$ to force the refining results have no nested spans. We use the grid search to search our hyperparameters, and the scope of each hyperparameter are included in Table 7. We train our model on a single NVIDIA A40 GPU with 48GB memory.

learning rate	[5e-5, 1e-4, 3e-4, 5e-4]
dropout	[0.1, 0.3, 0.5]
bert learning rate	[5e-6, 1e-5, 2e-5, 3e-5, 5e-5]
span dimension	[50, 100, 150, 200]
beam size	[1, 3, 5, 7]

Table 7: The searching scope of hyperparameters.

B FewNERD-nested

k	[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]
δ	[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]
u	[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]

Table 8: The searching scope of BSNMS hyperparameters.

We construct our FewNERD-nested via ACE05 (Walker et al., 2006), GENIA (Ohta et al., 2002) and the origin FewNERD datasets. GENIA is a biological named entity recognition dataset, which contains 5 kinds of entities, *'DNA'*, *'Protein'*, *'cell_type'*, *'RNA'* and *'cell_line'*, and all of these entity types are not included in the FewNERD. We

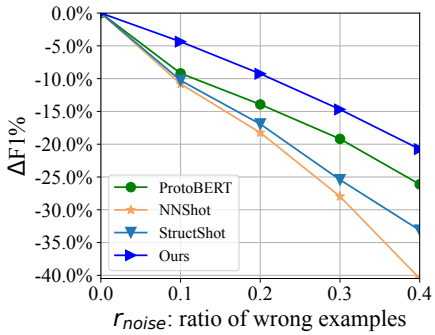

Figure 5: The performance drops in noisy situations when increasing r_{noise}. ΔF1% is the F1 reduction percent over $r_{noise} = 0$.

partition the sentences in GENIA into two groups, $\mathcal{G}_1$ and $\mathcal{G}_2$. $\mathcal{G}_1$ consists of sentences with nested entities ($4,744$ sentences in total), and $\mathcal{G}_2$ consists of sentences without nested entities ($11,924$ sentences in total). We utilize sentences in $\mathcal{G}_1$ and $\mathcal{G}_2$ to construct the query and support set of the GENIA task, respectively. Our FewNERD-nested contains 2000 5-way 5$\sim$10-shot test tasks in total, where r_{nested} percent tasks are from GENIA, and the remained tasks are from FewNERD-INTRA. We use ACE05 to construct the validation set. ACE05 is a widely used named entity recognition dataset, which contains 7 coarse-grained entity types, *'FAC'*, *'PER'*, *'LOC'*, *'VEH'*, *'GPE'*, *'GPE'*, *'WEA'* and *'ORG'*. The *'LOC'*, *'PER'*, *'ORG'* and *'FAC'* are not included in the training set of FewNERD-INTRA, and therefore we use them (28 fine-grained entity types in total) and sample 1000 nested tasks as the validation dataset to tune the k, δ and u of BSNMS. The search scope is included in the Table 8. In this nested situation, we finally set the k, δ and u to 0.1, 0.1 and 0.4 respectively.

C Robustness in the Noisy Situation

FSSL methods tend to be seriously influenced by the noise in the support set, since they make the prediction based on only limited annotated examples. To explore the robustness of ESD in the noisy situation, we construct **FewNERD-noise**. In FewNERD-noise, we disturb each 5 way 5 $\sim$ 10 shot FewNERD-INTRA task with a noisy ratio r_{noise}, which means there are nearly r_{noise} percent entities in the support set are mislabeled. As illustrated in the right part of Figure 5, with r_{noise} increasing, the performance of ESD drops less than baselines, which furthuer shows the superiority of our methods.

D A Detail Case Study of BSNMS

For span conflicts resolution, we propose a post-processing method BSNMS. A step by step demonstration of BSNMS is shown in Figure 6.

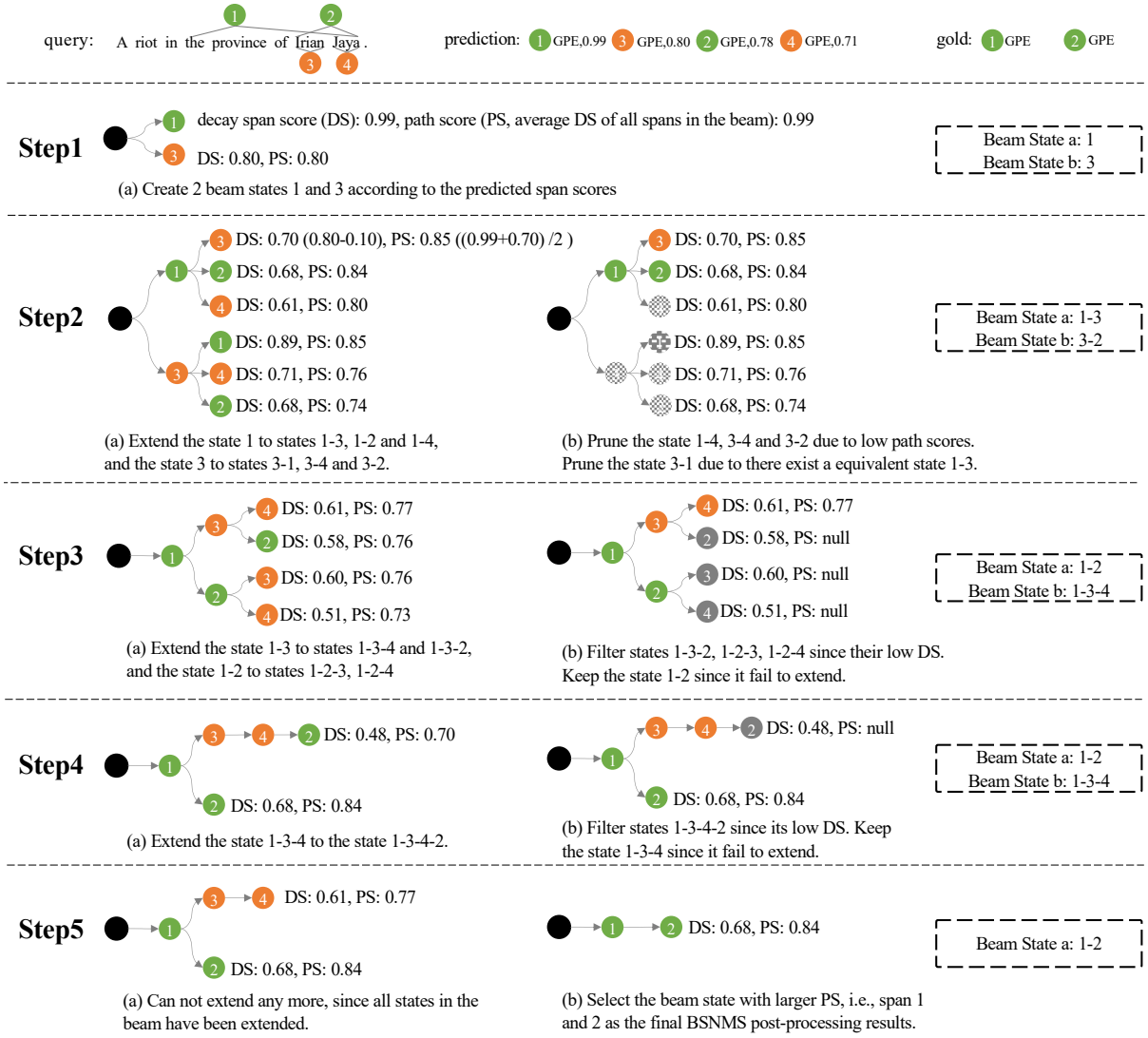

Figure 6: A step by step processing process of BSNMS with beam size=2. For clarity, we set the filter threshold δ to 0.6, and suppose the span score is always decayed by the overlapped spans with a constant decayed score -0.1. **STEP1**:create 2 (beam size) states with spans having larger predicted scores; **STEP2**: extend all states (i.e., states 1 and 3) in the beam to 1-3, 1-2, 1-4, 3-1, 3-4 and 3-2. Compute DS of the new added span and PS of the new state. As beam size=2, prune states 1-4, 3-4 and 3-2 according to their lower PS. The state 3-1 is dropped since it is equivalent with the state 1-3; **STEP3**: extend states in the beam (1-3 and 1-2) to 1-3-4, 1-3-2, 1-2-3, 1-2-4, and compute their DS and PS. Filter states 1-3-2, 1-2-3, 1-2-4 since their DS are not greater than δ, i.e., low DS; **STEP4**: extend states in the beam (only 1-3-4 since the state 1-2 has been extended before) to 1-3-4-2, and filter 1-3-4-2 due to its low DS; **STEP5**: all states in the beam can not extend any more, and select the final state (1-2 in this case) with the largest PS.