Parallel Instance Query Network for Named Entity Recognition

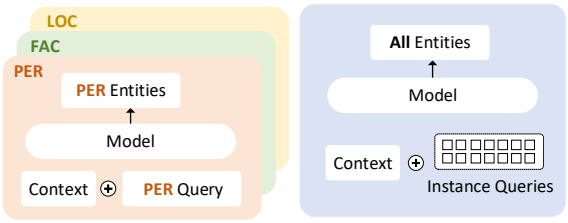
Yongliang Shen[1*], **Xiaobin Wang**[2], **Zeqi Tan**[1], **Guangwei Xu**[2],
Pengjun Xie[2], **Fei Huang**[2], **Weiming Lu**[1†], **Yueting Zhuang**[1]
[1]College of Computer Science and Technology, Zhejiang University
[2]DAMO Academy, Alibaba Group
{syl, luwm}@zju.edu.cn
xuanjie.wxb@alibaba-inc.com

Abstract

Named entity recognition (NER) is a fundamental task in natural language processing. Recent works treat named entity recognition as a reading comprehension task, constructing type-specific queries manually to extract entities. This paradigm suffers from three issues. First, type-specific queries can only extract one type of entities per inference, which is inefficient. Second, the extraction for different types of entities is isolated, ignoring the dependencies between them. Third, query construction relies on external knowledge and is difficult to apply to realistic scenarios with hundreds of entity types. To deal with them, we propose Parallel Instance Query Network (PIQN), which sets up global and learnable instance queries to extract entities from a sentence in a parallel manner. Each instance query predicts one entity, and by feeding all instance queries simultaneously, we can query all entities in parallel. Instead of being constructed from external knowledge, instance queries can learn their different query semantics during training. For training the model, we treat label assignment as a one-to-many Linear Assignment Problem (LAP) and dynamically assign gold entities to instance queries with minimal assignment cost. Experiments on both nested and flat NER datasets demonstrate that our proposed method outperforms previous state-of-the-art models[1].

1 Introduction

Named Entity Recognition (NER) aims to identify text spans to specific entity types such as Person, Location, Organization. It has been widely used in many downstream applications such as entity linking (Ganea and Hofmann, 2017; Le and Titov, 2018) and relation extraction (Li and Ji, 2014; Miwa and Bansal, 2016; Shen et al., 2021b). Traditional approaches for NER are based on sequence labeling, assigning a single tag to each word in a sentence. However, the words of nested entities have more than one tag, thus these methods lack the ability to identify nested entities.



(a) Type-specific Query (b) Instance Query

Figure 1: (a) For a sentence, type-specific queries can only extract entities of one type per inference, so the model needs to be run multiple times. (b) In contrast, instance-based queries can be input into the model simultaneously, and all entities can be extracted in parallel. Furthermore, the parallel manner can model the interactions between entities of different types.

Recently, Ju et al. (2018); Straková et al. (2019); Wang et al. (2020a) redesign sequence labeling models to support nested structures using different strategies. Instead of labeling each word, Luan et al. (2019); Tan et al. (2020); Li et al. (2021); Shen et al. (2021a) perform a classification task on the text span, and Straková et al. (2019); Paolini et al. (2021); Yan et al. (2021); Tan et al. (2021) treat NER as a sequence generation or set prediction task and design encoder-decoder models to generate entities. Recently, Li et al. (2020b); Mengge et al. (2020); Zheng et al. (2021) reformulate the NER task as a machine reading task and achieve a promising performance on both flat and nested datasets. As shown in Figure 1(a), they treat the sentence as context and construct type-specific queries from external knowledge to extract entities. For example, for the sentence *"U.S. President Barack Obama and his wife spent eight years*

[*] This work was conducted when Yongliang Shen was interning at Alibaba DAMO Academy.
[†] Corresponding author.
[1] Our code is available at https://github.com/tricktreat/piqn.

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 947 - 961

in the White House", Li et al. (2020b) constructs the PER-specific query in natural language form - *"Find person entity in the text, including a single individual or a group"* to extract the PER entities, such as *"U.S. President"*, *"Barack Obama"*. However, since the queries are type-specific, only one type of entities can be extracted for each inference. This manner not only leads to inefficient prediction but also ignores the intrinsic connections between different types of entities, such as *"U.S."* and *"U.S. President"*. In addition, type-specific queries rely on external knowledge for manual construction, which makes it difficult to fit realistic scenarios with hundreds of entity types.

In this paper, we propose the Parallel Instance Query Network (PIQN), where global and learnable instance queries replace type-specific ones to extract entities in parallel. As shown in Figure 1(b), each instance query predicts one entity, and multiple instance queries can be fed simultaneously to predict all entities. Different from previous methods, we do not need external knowledge to construct the query into natural language form. The instance query can learn different query semantics during training, such as position-related or type-related semantics. Since the semantics of instance queries are implicit, we cannot assign gold entities as their labels in advance. To tackle this, we treat label assignment as a one-to-many Linear Assignment Problem (LAP) (Burkard and Çela, 1999), and design a dynamic label assignment mechanism to assign gold entities for instance queries.

Our main contributions are as follow:

- Different from type-specific queries that require multiple rounds of query, our model employs instance queries that can extract all entities in parallel. Furthermore, the style of parallel query can model the interactions between entities of different types.
- Instead of relying on external knowledge to construct queries in natural language form, instance queries learn their query semantics related to entity location and entity type during training.
- To train the model, we design a dynamic one-to-many label assignment mechanism, where the entities are dynamically assigned as labels for the instance queries during training. The one-to-many manner allows multiple queries to predict the same entity, which can further improve the model performance.
- Experiments show that our model achieves state-of-the-art performance consistently on several nested and flat NER datasets.

2 Related Work

Traditional approaches for NER can be divided into three categories, including tagging-based, hypergraph-based and span-based approaches. The typical sequence labeling approach (Huang et al., 2015) predicts labels for each token, and struggles to address nested NER. Some works (Alex et al., 2007; Wang et al., 2020a) adapt the sequence labeling model to nested entity structures by designing a special tagging scheme. Different from the decoding on the linear sequence, the hypergraph-based approaches (Lu and Roth, 2015; Muis and Lu, 2017; Katiyar and Cardie, 2018) construct hypergraphs based on the entity nesting structure and decode entities on the hypergraph. Span-based methods first extract spans by enumeration (Sohrab and Miwa, 2018; Luan et al., 2019) or boundary identification (Zheng et al., 2019; Tan et al., 2020), and then classify the spans. Based on these, Shen et al. (2021a) treats NER as a joint task of boundary regression and span classification and proposes a two-stage identifier of locating entities first and labeling them later.

Three novel paradigms for NER have recently been proposed, reformulating named entity recognition as sequence generation, set prediction, and reading comprehension tasks, respectively. Yan et al. (2021) formulates NER as an entity span sequence generation problem and uses a BART (Lewis et al., 2020) model with the pointer mechanism to tackle NER tasks. Tan et al. (2021) formulates NER as an entity set prediction task. Different from Straková et al. (2019), they utilize a non-autoregressive decoder to predict entity set. Li et al. (2020b); Mengge et al. (2020) reformulate the NER task as an MRC question answering task. They construct type-specific queries using semantic prior information for entity categories.

Different from Li et al. (2020b); Jiang et al. (2021), our method attempts to query at the entity level, where it adaptively learns query semantics for instance queries and extracts all types of entities in parallel. It is worth noting that Seq2Set (Tan et al., 2021) is quite different from ours: (1)

Seq2Set attempts to eliminate the incorrect bias introduced by specified entity decoding order in the seq2seq framework, and proposes an entity set predictor, while we follow the MRC paradigm and focus on extracting entities using instance queries. (2) Seq2Set is an encoder-decoder architecture, while our model throws away the decoder and keeps only the encoder as in Wang et al. (2022a), which speeds up inference and allows full interaction between query and context. (3) Seq2Set uses bipartite graph matching to compute the entity-set level loss, while we focus on the label assignment for each instance query and propose a one-to-many dynamic label assignment mechanism.

3 Method

In this section, we first introduce the task formulation in § 3.1, and then describe our method. As shown in Figure 2, our method consists of three components: the Encoder (§ 3.2), the Entity Prediction (§ 3.3) and the Dynamic Label Assignment (§ 3.4). The encoder encodes both the sentence and instance queries. Then for each instance query, we perform entity localization and entity classification using Entity Pointer and Entity Classifier respectively. For training the model, we introduce a dynamic label assignment mechanism to assign gold entities to the instance queries in § 3.4.

3.1 Task Formulation

We use (X, Y) to denote a training sample, where X is a sentence consisting of N words labeled by a set of triples $Y = \{< Y_k^l, Y_k^r, Y_k^t >\}_{k=0}^{G-1}$. $Y_k^l \in [0, N-1]$, $Y_k^r \in [0, N-1]$ and $Y_k^t \in \mathcal{E}$ are the indices for the left boundary, right boundary and entity type of the k-th entity, where $\mathcal{E}$ is a finite set of entity types. In our approach, We set up $M (M > G)$ global and learnable instance queries $I = \mathbb{R}^{M \times h}$, each of which (denoted as a vector of size h) extracts one entity from the sentence. They are randomly initialized and can learn the query semantics automatically during training. Thus we define the task as follows: given an input sentence X, the aim is to extract the entities Y based on the learnable instance queries I.

3.2 Encoder

Model input consists of two sequences, the sentence X of length N and the instance queries I of length M. The encoder concatenates them into one sequence and encodes them simultaneously.

Input Embedding We calculate the token embeddings E_{tok}, position embeddings E_{pos} and type embeddings E_{typ} of the input from two sequences as follows ($E_{tok}, E_{pos}, E_{typ} \in \mathbb{R}^{(N+M) \times h}$):

$$\begin{aligned} E_{tok} &= \text{Concat}(V, I) \\ E_{pos} &= \text{Concat}(P^w, P^q) \\ E_{typ} &= \text{Concat}([U^w]^N, [U^q]^M) \end{aligned} \tag{1}$$

where $V \in \mathbb{R}^{N \times h}$ are token embeddings of the word sequence, $I \in \mathbb{R}^{M \times h}$ are the vectors of instance queries, $P^w \in \mathbb{R}^{N \times h}$ and $P^q \in \mathbb{R}^{M \times h}$ are separate learnable position embeddings. U^w and U^q are type embeddings and $[\cdot]^N$ means repeating N times. Then the input can be represented as $H^0 = E_{tok} + E_{pos} + E_{typ} \in R^{(N+M) \times h}$.

One-Way Self-Attention Normal self-attention would let the sentence interact with all instance queries. In such a way, randomly initialized instance queries can affect the sentence encoding and break the semantics of the sentence. To keep the sentence semantics isolated from the instance queries, we replace the self-attention in BERT (Devlin et al., 2019) with the one-way version:

$$\text{OW-SA}(H) = \alpha H W_v \tag{2}$$

$$\alpha = \text{softmax}\left(\frac{H W_q (H W_k)^T}{\sqrt{h}} + \mathcal{M}\right) \tag{3}$$

where $W_q, W_k, W_v \in \mathbb{R}^{h \times h}$ are parameter matrices and $\mathcal{M} \in \{0, -\inf\}^{(N+M) \times (N+M)}$ is a mask matrix for the attention score where elements in $\mathcal{M}$ set to 0 for kept units and $-\inf$ for removed ones. In our formula, the upper right sub-matrix of $\mathcal{M}$ is a full $-\inf$ matrix of size $(N \times M)$ and other elements are zero, which can prevent the sentence encoding from attending on the instance queries. In addition, the self-attention among instance queries can model the connections between each other, and then enhance their query semantics.

After BERT encoding, we further encode the sequence at word-level by two bidirectional LSTM layers and L extra transformer layers. Finally we split $H \in \mathbb{R}^{(N+M) \times h}$ into two parts: the sentence encoding $H^w \in \mathbb{R}^{N \times h}$ and the instance query encoding $H^q \in \mathbb{R}^{M \times h}$.

3.3 Entity Prediction

Each instance query can predict one entity from the sentence, and with M instance queries, we can predict at most M entities in parallel. Entity prediction

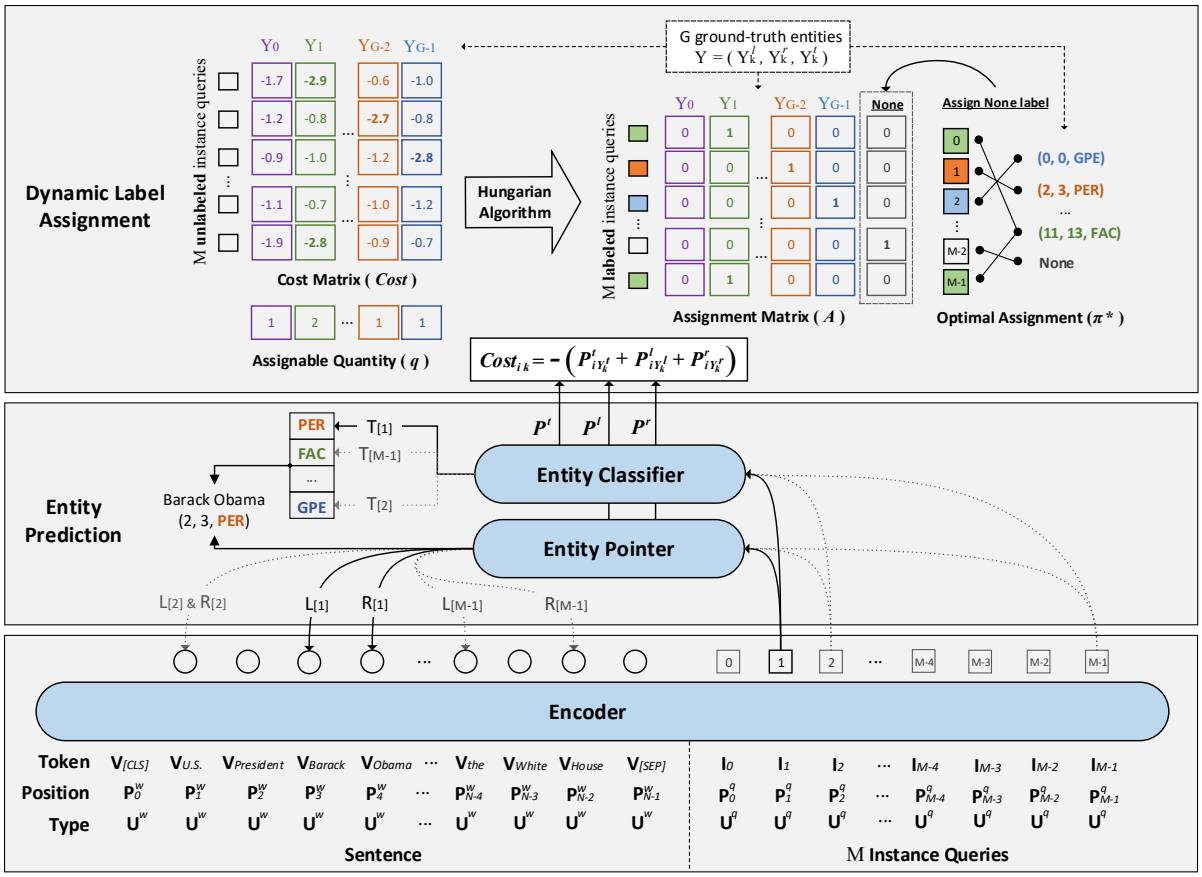


Figure 2: The overall architecture of the model.

can be viewed as a joint task of boundary prediction and category prediction. We design Entity Pointer and Entity Classifier for them respectively.

Entity Pointer For the i-th instance query H_i^q, we first interact the query with each word of the sentence by two linear layers. The fusion representation of the i-th instance query and j-th word is computed as:

$$S_{ij}^{\delta} = \text{ReLU}(H_i^q W_{\delta}^q + H_j^w W_{\delta}^w) \tag{4}$$

where $\delta \in \{l, r\}$ denotes the left or right boundary and $W_{\delta}^q, W_{\delta}^w \in \mathbb{R}^{h \times h}$ are trainable projection parameters. Then we calculate the probability that the j-th word of the sentence is a left or right boundary:

$$P_{ij}^{\delta} = \text{sigmoid}(S_{ij}^{\delta} W_{\delta} + b_{\delta}) \tag{5}$$

where $W_{\delta} \in \mathbb{R}^h$ and b_{δ} are learnable parameters.

Entity Classifier Entity boundary information are useful for entity typing. We use $P_i^{\delta} = [P_{i0}^{\delta}, P_{i1}^{\delta}, \cdots, P_{iN-1}^{\delta}], \delta \in \{l, r\}$ to weigh all words and then concatenate them with instance queries. The boundary-aware representation of the i-th instance query can be calculated as:

$$S_i^t = \text{ReLU}\left(\left[H_i^q W_t^q; P_i^l H^w; P_i^r H^w\right]\right) \tag{6}$$

where $W_t^q \in \mathbb{R}^{h \times h}$ is a learnable parameter. Then we can get the probability of the entity queried by the i-th instance query belonging to category c:

$$P_{ic}^t = \frac{\exp(S_i^t W_t^c + b_t^c)}{\sum_{c' \in \mathcal{E}} \exp(S_i^t W_t^{c'} + b_t^{c'})} \tag{7}$$

where $W_t^{c'} \in \mathbb{R}^h$ and $b_t^{c'}$ are learnable parameters.

Finally, the entity predicted by the i-th instance query is $\mathcal{T}_i = (\mathcal{T}_i^l, \mathcal{T}_i^r, \mathcal{T}_i^t)$. $\mathcal{T}_i^l = \arg\max_j(P_{ij}^l)$ and $\mathcal{T}_i^r = \arg\max_j(P_{ij}^r)$ are the left and right boundary, $\mathcal{T}_i^t = \arg\max_c(P_{ic}^t)$ is the entity type. We perform entity localization and entity classification on all instance queries to extract entities in parallel. If multiple instance queries locate the same entity but predict different entity types, we keep only the prediction with the highest classification probability.

3.4 Dynamic Label Assignment for Training

Dynamic Label Assignment Since instance queries are implicit (not in natural language form), we cannot assign gold entities to them in advance. To tackle this, we dynamically assign labels for the instance queries during training. Specifically, we treat label assignment as a Linear Assignment Problem. Any entity can be assigned to any instance query, incurring some cost that may vary depending on the entity-query assignment. We define the cost of assigning the k-th entity ($Y_k = < Y_k^l, Y_k^r, Y_k^t >$) to the i-th instance query as:

$$Cost_{ik} = -\left(P^t_{iY^t_k} + P^l_{iY^l_k} + P^r_{iY^r_k}\right) \tag{8}$$

where Y_k^t, Y_k^l and Y_k^r denote the indices for the entity type, left boundary and right boundary of the k-th entity. It is required to allocate as many entities as possible by assigning at most one entity to each query and at most one query to each entity, in such a way that the total cost of the assignment is minimized. However, the one-to-one manner does not fully utilize instance queries, and many instance queries are not assigned to gold entities. Thus we extend the traditional LAP to one-to-many one, where each entity can be assigned to multiple instance queries. The optimization objective of this one-to-many LAP is defined as:

$$\begin{aligned} \min & \sum_{i=0}^{M-1}\sum_{k=0}^{G-1} A_{ik} Cost_{ik} \\ s.t. \quad & \textstyle\sum_k A_{ik} \le 1 \\ & \textstyle\sum_i A_{ik} = q_k \\ & \forall i,k, A_{ik} \in \{0,1\} \end{aligned} \quad . \tag{9}$$

where $A \in \{0,1\}^{M\times G}$ is the assignment matrix, G denotes the number of the entities and A_{ik} = 1 indicates the k-th entity assigned to the i-th instance query. q_k denotes the assignable quantity of the k-th gold entity and $Q = \sum_k q_k$ denotes the total assignable quantity for all entities. In our experiments, the assignable quantities of different entities are balanced.

We then use the Hungarian (Kuhn, 1955) algorithm to solve Equation 9, which yields the label assignment matrix with the minimum total cost. However, the number of instance queries is greater than the total assignable quantity of entity labels ($M > Q$), so some of them will not be assigned to any entity label. We assign `None` label to them by extending a column for the assignment matrix. The new column vector a is set as follows:

$$a_i = \begin{cases} 0, & \sum_k A_{ik} = 1 \\ 1, & \sum_k A_{ik} = 0 \end{cases} \tag{10}$$

Based on the new assignment matrix $\hat{A} \in \{0,1\}^{M\times(G+1)}$, we can further get the labels $\hat{Y} = Y.\text{indexby}(\pi^*)$ for M instance queries, where $\pi^* = \arg\max_{dim=1}(\hat{A})$ is the label index vector for instance queries under the optimal assignment.

Training Objective We have computed the entity predictions for M instance queries in § 3.3 and got their labels $\hat{Y}$ with the minimum total assignment cost in § 3.4. To train the model, we define boundary loss and classification loss. For left and right boundary prediction, we use binary cross entropy function as a loss:

$$\begin{aligned} \mathcal{L}_b = -\sum_{\delta\in\{l,r\}}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1} & \mathbb{1}[\hat{Y}_i^\delta = j]\log P_{ij}^\delta \\ & + \mathbb{1}[\hat{Y}_i^\delta \neq j]\log\left(1 - P_{ij}^\delta\right) \end{aligned} \tag{11}$$

and for entity classification we use cross entropy function as a loss:

$$\mathcal{L}_t = -\sum_{i=0}^{M-1}\sum_{c\in\mathcal{E}} \mathbb{1}[\hat{Y}_i^t = c]\log P_{ic}^t \tag{12}$$

where $\mathbb{1}[\omega]$ denotes indicator function that takes 1 when ω is true and 0 otherwise.

Follow Al-Rfou et al. (2019) and Carion et al. (2020), we add Entity Pointer and Entity Classifier after each word-level transformer layer, and we can get the two losses at each layer. Thus, the total loss on the train set D can be defined as:

$$\mathcal{L} = \sum_D \sum_{\tau=1}^{L} \mathcal{L}_t^\tau + \mathcal{L}_b^\tau \tag{13}$$

where $\mathcal{L}_t^\tau, \mathcal{L}_b^\tau$ are classification loss and boundary loss at the τ-th layer. For prediction, we just perform entity prediction at the final layer.

4 Experiment Settings

4.1 Datasets

To provide empirical evidence for the effectiveness of the proposed model, we conduct our experiments

Model	ACE04		
	Pr.	Rec.	F1
Li et al. (2020b)	85.05	86.32	85.98
Wang et al. (2020a)	86.08	86.48	86.28
Yu et al. (2020)	87.30	86.00	86.70
Yan et al. (2021)	87.27	86.41	86.84
Yang and Tu (2022)	86.60	87.28	86.94
Tan et al. (2021)	88.46	86.10	87.26
Shen et al. (2021a)	87.44	87.38	87.41
PIQN	**88.48**	**87.81**	**88.14**
Model	ACE05		
	Pr.	Rec.	F1
Lin et al. (2019)	76.20	73.60	74.90
Luo and Zhao (2020)	75.00	75.20	75.10
Li et al. (2021)	-	-	83.00
Wang et al. (2020a)	83.95	85.39	84.66
Yan et al. (2021)	83.16	86.38	84.74
Yu et al. (2020)	85.20	85.60	85.40
Yang and Tu (2022)	84.61	86.43	85.53
Li et al. (2020b)	87.16	86.59	86.88
Shen et al. (2021a)	86.09	87.27	86.67
Tan et al. (2021)	87.48	86.63	87.05
PIQN	86.27	**88.60**	**87.42**
Model	GENIA		
	Pr.	Rec.	F1
Lin et al. (2019)	75.80	73.90	74.80
Luo and Zhao (2020)	77.40	74.60	76.00
Wang et al. (2020b)	78.10	74.40	76.20
Yang and Tu (2022)	78.08	78.26	78.16
Li et al. (2020b)†	81.14	76.82	78.92
Wang et al. (2020a)	79.45	78.94	79.19
Yan et al. (2021)	78.87	79.6	79.23
Tan et al. (2021)	82.31	78.66	80.44
Yu et al. (2020)	81.80	79.30	80.50
Shen et al. (2021a)	80.19	80.89	80.54
PIQN	**83.24**	80.35	**81.77**
Model	KBP17		
	Pr.	Rec.	F1
Ji et al. (2017)	76.20	73.00	72.80
Lin et al. (2019)	77.70	71.80	74.60
Luo and Zhao (2020)	77.10	74.30	75.60
Li et al. (2020b)	80.97	81.12	80.97
Tan et al. (2021)	84.91	83.04	83.96
Shen et al. (2021a)	85.46	82.67	84.05
PIQN	**85.67**	**83.37**	**84.50**
Model	NNE		
	Pr.	Rec.	F1
Li et al. (2020b)‡	53.13	56.67	54.84
Wang and Lu (2018)	77.40	70.10	73.60
Ringland et al. (2019)	91.80	91.00	91.40
Tan et al. (2021)‡	93.01	89.21	91.07
Shen et al. (2021a)‡	92.86	91.12	91.98
Wang et al. (2020a)†	92.64	93.53	93.08
PIQN	**93.85**	**94.23**	**94.04**

Table 1: Results for ***nested*** NER task. † means the reproduction on the same preprocessed dataset and ‡ means that we run the code on the unreported dataset.

on eight English datasets, including five nested NER datasets: ACE04 (Doddington et al., 2004) , ACE05 (Walker et al., 2006), KBP17 (Ji et al., 2017), GENIA (Ohta et al., 2002), NNE(Ringland et al., 2019) and three flat NER dataset: FewNERD (Ding et al., 2021), CoNLL03 (Tjong Kim Sang and De Meulder, 2003), OntoNotes (Pradhan et al., 2013), and one Chinese flat NER dataset: MSRA (Levow, 2006). FewNERD and NNE are two datasets with large entity type inventories, containing 66 and 114 fine-grained entity types. Please refer to Appendix A for statistical information about the datasets.

4.2 Implementation Details

In our experiments, we use pretrained BERT (Devlin et al., 2019) in our encoder. For a fair comparison, we use `bert-large` on ACE04, ACE05, NNE, CoNLL03 and OntoNotes, `bert-base` on KBP17 and FewNERD, `biobert-large` (Chiu et al., 2016) on GENIA and `chinese-bert-wwm` (Cui et al., 2020) on Chinese MSRA. For all datasets, we train our model for 30-60 epochs and use the Adam Optimizer (Kingma and Ba, 2015) with a linear warmup-decay learning rate schedule. We initialize all instance queries using the normal distribution $\mathcal{N}(0.0, 0.02)$. See Appendix B for more detailed parameter settings and Appendix C for all baseline models.

4.3 Evaluation Metrics

We use strict evaluation metrics that an entity is confirmed correct when the entity boundary and the entity type are correct simultaneously. We employ precision, recall and F1-score to evaluate the performance. We also report the F1-scores on the entity localization and entity classification subtasks in § 5.2 and Appendix D.2. We consider the localization as correct when the left and right boundaries are predicted correctly. Based on the accurately localized entities, we then evaluate the performance of entity classification.

5 Results and Analysis

5.1 Performance

Overall Performance Table 1 illustrates the performance of the proposed model as well as baselines on the nested NER datasets. We observe significant performance boosts on the nested NER datasets over previous state-of-the-art models,

achieving F1-scores of 81.77%, 88.14%, 87.42% and 84.50% on GENIA, ACE04, ACE05, KBP17 and NNE datasets with +1.23%, +0.73%, +0.37%, +0.45% and +0.96% improvements. Our model can be applied to flat NER. As shown in Table 2, our model achieves state-of-the-art performance on the FewNERD and Chinese MSRA datasets with +1.44% and +0.88% improvements. On the CoNLL03 and OntoNotes datasets, our model also achieves comparable results. Compared with the type-specific query-based method (Li et al., 2020b), our model improves by +2.85%, +2.16%, +0.54%, +3.53% on the GENIA, ACE04, ACE05 and KBP17 datasets. We believe there are three reasons: (1) Rather than relying on external knowledge to inject semantics, instance queries can learn query semantics adaptively, avoiding the sensitivity to hand-constructed queries of varying quality. (2) Each query no longer predicts a group of entities of a specific type, but only one entity. This manner refines the query to the entity level with more precise query semantics. (3) Instance queries are fed into the model in parallel for encoding and prediction, and different instance queries can exploit the intrinsic connections between entities.

Inference Speed We compare the inference speed on ACE04 and NNE, as shown in Table 4. Compared to the type-specific query method (Li et al., 2020b), our model not only improves the performance, but also gains significant inference speedup. In particular, on the NNE dataset with 114 entity types, our model speeds up by 30.46× and improves performance by +39.2%. This is because Li et al. (2020b) requires one inference for each type-specific query, while our approach performs parallel inference for all instance queries and only needs to be run once. We also compare previous state-of-the-art models (Tan et al., 2021; Shen et al., 2021a) and our method is still faster and performs better.

Model	FewNERD		
	Pr.	Rec.	F1
Ding et al. (2021)	65.56	68.78	67.13
Shen et al. (2021a)‡	64.69	70.87	67.64
Tan et al. (2021)‡	67.37	69.12	68.23
PIQN	**70.16**	**69.18**	**69.67**
Model	English CoNLL03		
	Pr.	Rec.	F1
Peters et al. (2018)	-	-	92.22
Devlin et al. (2019)	-	-	92.80
Li et al. (2020b)∗	92.47	93.27	92.87
Yu et al. (2020)∗	92.85	92.15	92.50
Shen et al. (2021a)	92.13	93.73	92.94
PIQN	**93.29**	92.46	92.87
Model	English OntoNotes		
	Pr.	Rec.	F1
Li et al. (2020b)∗	91.34	88.39	89.84
Yu et al. (2020)∗	89.74	89.92	89.83
Yan et al. (2021)	89.99	90.77	90.38
Xu et al. (2021)	90.14	91.58	90.85
PIQN	**91.43**	90.73	**90.96**
Model	Chinese MSRA		
	Pr.	Rec.	F1
Devlin et al. (2019)	-	-	92.60
Li et al. (2020b)†	90.38	89.00	89.68
Shen et al. (2021a)‡	92.20	90.72	91.46
Tan et al. (2021)‡	93.21	91.97	92.58
PIQN	**93.61**	**93.35**	**93.48**

Table 2: Results for ***flat*** NER task. ∗ means the result reproduced by (Yan et al., 2021), † means the reproduction on the same preprocessed dataset and ‡ means that we run the code on the unreported dataset.

5.2 Ablation Study

In this section, we analyze the effects of different components in PIQN. As shown in Table 3, we have the following observations: (1) Compared to the static label assignment in order of occurrence, the dynamic label assignment shows significant improvement on localization, classification, and NER F1-score, which improves NER F1-score by +5.71% on ACE04 and +8.84% on GENIA. This shows that modeling label assignment as a LAP problem enables dynamic assignment of optimal labels to instance queries during training, eliminating the incorrect bias when pre-specifying labels. Furthermore, one-to-many for label assignment is more effective than one-to-one, improving the F1-score by +3.86% on ACE04 and +0.51% on GENIA. (2) The one-way self-attention blocks the attention of sentence encoding on instance queries, which improves the F1-score by +0.98% on ACE04 and +0.57% on GENIA. It illustrates the importance of keeping the semantics of the sentence independent of the query. In contrast, semantic interactions between queries are effective, which improves the F1-score by +0.92% on ACE04 and +0.67% on GENIA. The major reason is that entities in the same sentence are closely related and the interaction between instance queries can capture the relation between them.

Model	ACE04					GENIA				
	Loc. F1	Cls. F1	Pr.	Rec.	F1	Loc. F1	Cls. F1	Pr.	Rec.	F1
Default	**92.23**	**91.53**	**88.48**	**87.81**	**88.14**	**84.43**	**87.83**	**83.24**	**80.35**	**81.77**
w/o Dynamic LA	88.22	88.29	80.95	83.99	82.43	77.01	81.90	73.56	72.30	72.93
w/o OvM LA	89.22	87.61	87.04	81.68	84.28	83.87	87.38	83.02	79.57	81.26
w/o One Way SA	91.90	90.62	87.56	86.75	87.16	84.11	87.21	82.94	79.53	81.20
w/o Query Interaction	91.84	90.42	88.21	86.26	87.22	83.87	87.05	83.15	79.15	81.10

Table 3: Ablation Study. (1) **w/o Dynamic LA**: replace dynamic label assignment to static label assignment, i.e., assign labels to instance queries in the order of the entities' occurrence in the sentence. (2) **w/o OvM LA**: replace the one-to-many label assignment to one-to-one, i.e., set the number of queries to which each entity can be assigned to be 1. (3) **w/o One Way SA**: encode sentences and instance queries using the original BERT. (4) **w/o Query Interaction**: eliminate interactions between instance queries by masking the attention weights between them.

Model	ACE04		NNE	
	Speedup	F1	Speedup	F1
Li et al. (2020b)	1.00×	85.98	1.00×	54.84
Tan et al. (2021)	1.40×	87.26	22.18×	91.07
Shen et al. (2021a)	0.96×	87.41	11.41×	91.98
PIQN	**2.16×**	**88.14**	**30.46×**	**94.04**

Table 4: Inference Speed on ACE04 and NNE. All experiments are conducted on a single NVIDIA RTX A6000 Graphical Card with 48G graphical memory.

5.3 Analysis

In order to analyze the query semantics learned by the instance query in the training, we randomly selected several instance queries and analyzed the locations and types of entities they predicted.

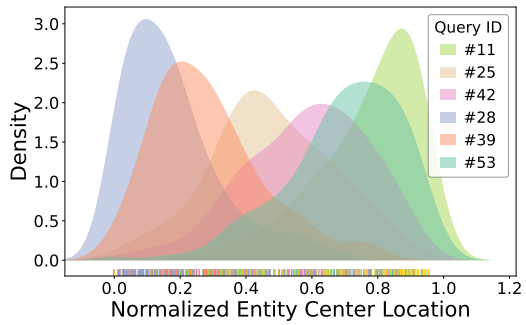


Figure 3: Kernel density estimation of entity distribution at different locations.

Entity Location We normalize the predicted central locations of the entities and use kernel density estimation to draw the distribution of the predicted entity locations for different queries, as shown in Figure 3. We observe that different instance queries focus on entities at different positions, which means that the instance queries can learn the query semantics related to entity position. For example, instance queries #28 and #39 prefer to predict entities at the beginning of sentences, while #11 and #53 prefer entities at the end.

Entity Type We count the co-occurrence of different instance queries and different entity types they predicted. To eliminate the imbalance of entity types, we normalize the co-occurrence matrix on the entity type axis. As shown in Figure 4, different instance queries have preferences for different entity types. For example, instance queries #11 and #13 prefer to predict PER entities, #30 and #43 prefer VEH entities, #25 and #49 prefer WEA entities, #12 prefers FAC entities, and #35 prefers LOC entities.

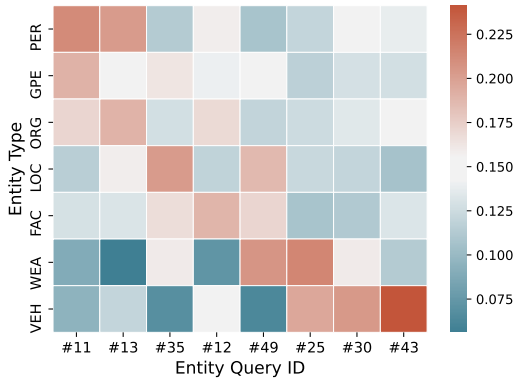


Figure 4: Co-occurrence statistics between instance queries and different entity types

We also analyze the auxiliary loss, the dynamic label assignment mechanism, and the performance on entity localization and classification, please see the Appendix D.

6 Case Study

Table 5 shows a case study about model predictions. Our model can recognize nested entities and

#	Sentence with Gold Entities	Prediction ← Instance Query IDs
1	[0A number of powerful international companies and commercial agencies , such as [12Ito Bureau of [15Japan15]GPE15]ORG , [17Han Hua Group of [21South Korea22]GPE22]ORG , [24Jeffrey Group of [27the US28]GPE28]ORG , [30etc30]ORG30]ORG . participated in this Urumchi Negotiation Meeting .	✓ (24, 28, ORG) ← 0 23 33 45 51 ✓ (27, 28, GPE) ← 2 3 19 26 27 46 50 ✓ (15, 15, GPE) ← 9 11 14 42 ✓ ... ... ✓ (0, 30, ORG) ← 10 20 24 37 53 55 ✗ (12, 30, ORG) ← 16 22 47 57 **None** ← 1 12 13 15 17 21 29 30 31 32 34 35 40 49 52 59
2	For example , as instant messaging migrates to cell phones or hand - held computer organizers , [17consumers17]PER won ' t want to have to install multiple services on these devices , said [33Brian Park34]PER , [36senior product for [39Yahoo !40]ORG Communications Services42]PER .	✗ (39, 42, ORG) ← 0 2 15 19 26 27 29 35 46 49 50 ✓ (17, 17, PER) ← 1 10 20 22 24 32 37 47 53 55 57 ✓ (33, 34, PER) ← 6 9 11 12 14 18 34 38 42 48 59 ✓ (36, 42, PER) ← 8 17 25 28 30 31 36 40 54 56 58 **None** ← 3 4 5 7 13 16 21 23 33 39 41 43 44 45 51 52
3	[0Hector Rodriguez1]PER told the hearing of [6the Venezuelan consumer protection agency10]ORG that [12Bridgeton Firestone13]ORG knew about the tyre defects for many months and should be held responsible for the accidents .	✓ (0, 1, PER) ← 1 10 20 24 32 37 47 53 55 ✓ (12, 13, ORG) ← 2 3 19 26 27 35 46 49 50 ✗ (7, 8, PER) ← 4 7 12 18 38 39 41 43 44 ✓ (6, 10, ORG) ← 5 6 9 11 14 21 48 57 59 ✗ (7, 7, GPE) ← 8 25 28 30 31 36 40 54 56 58 **None** ← 0 13 15 16 17 22 23 29 33 34 42 45 51 52

Table 5: Cases Study. In the left column, the label in the lower right corner indicates the type of entity, and the superscripts indicate the positions of the left and right boundary words. In the right column, we show the correspondence between the instance queries and the predicted entities.

long entities well. In case 1, the entities of length 31 or with the three-level nested structure are predicted accurately. And thanks to the one-to-many dynamic label assignment mechanism, each entity can be predicted by multiple instance queries, which guarantees a high coverage of entity prediction. However, the model's ability to understand sentences is still insufficient, mainly in the following ways: (1) There is a deficiency in the understanding of special phrases. *Yahoo ! Communications Services* in case 2 is misclassified as `ORG`, but in fact *Yahoo !* is `ORG`. (2) Over-focus on local semantics. In case 3, the model misclassifies *Venezuelan consumer* as `PER`, ignoring the full semantics of the long phrase *the Venezuelan consumer protection agency*, which should be `ORG`. (3) Insensitivity to morphological variation. The model confused *Venezuelan* and *Venezuela*, and misidentified the former as `GPE` in case 3.

7 Conclusion

In this paper, we propose Parallel Instance Query Network for nested NER, where a collection of instance queries are fed into the model simultaneously and can predict all entities in parallel. The instance queries can automatically learn query semantics related to entity types or entity locations during training, avoiding manual constructions that rely on external knowledge. To train the model, we design a dynamic label assignment mechanism to assign gold entities for these instance queries. Experiments on both nested and flat NER datasets demonstrate that the proposed model achieves state-of-the-art performance.

Acknowledgments

This work is supported by the Key Research and Development Program of Zhejiang Province, China (No. 2021C01013), the National Key Research and Development Project of China (No. 2018AAA0101900), the Chinese Knowledge Center of Engineering Science and Technology (CKCEST) and MOE Engineering Research Center of Digital Library.

References

Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. 2019. Character-level language modeling with deeper self-attention. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:3159–3166.

Beatrice Alex, Barry Haddow, and Claire Grover. 2007. Recognising nested named entities in biomedical text. In *Biological, translational, and clinical language processing*, pages 65–72, Prague, Czech Republic. Association for Computational Linguistics.

R. Burkard and E. Çela. 1999. Linear assignment problems and extensions. In *Handbook of Combinatorial Optimization*.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-end object detection with transformers. In *ECCV 2020*, pages 213–229, Cham. Springer International Publishing.

Billy Chiu, Gamal Crichton, Anna Korhonen, and Sampo Pyysalo. 2016. How to train good word embeddings for biomedical NLP. In *Proceedings of the 15th Workshop on Biomedical Natural Language Processing*, pages 166–174, Berlin, Germany. Association for Computational Linguistics.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Shijin Wang, and Guoping Hu. 2020. Revisiting pre-trained models for Chinese natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 657–668, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Ning Ding, Guangwei Xu, Yulin Chen, Xiaobin Wang, Xu Han, Pengjun Xie, Haitao Zheng, and Zhiyuan Liu. 2021. Few-NERD: A few-shot named entity recognition dataset. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, pages 3198–3213, Online. Association for Computational Linguistics.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The automatic content extraction (ACE) program – tasks, data, and evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal. European Language Resources Association (ELRA).

Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep joint entity disambiguation with local neural attention. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2619–2629, Copenhagen, Denmark. Association for Computational Linguistics.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Heng Ji, Xiaoman Pan, Boliang Zhang, Joel Nothman, James Mayfield, Paul McNamee, and Cash Costello. 2017. Overview of TAC-KBP2017 13 languages entity discovery and linking. In *Proceedings of the 2017 Text Analysis Conference, TAC 2017, Gaithersburg, Maryland, USA, November 13-14, 2017*. NIST.

Xiaobo Jiang, Kun He, Jiajun He, and Guangyu Yan. 2021. A new entity extraction method based on machine reading comprehension.

Meizhi Ju, Makoto Miwa, and Sophia Ananiadou. 2018. A neural layered model for nested named entity recognition. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1446–1459, New Orleans, Louisiana. Association for Computational Linguistics.

Arzoo Katiyar and Claire Cardie. 2018. Nested named entity recognition revisited. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 861–871, New Orleans, Louisiana. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3th International Conference on Learning Representations, ICLR 2021*.

Harold W Kuhn. 1955. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

Phong Le and Ivan Titov. 2018. Improving entity linking by modeling latent relations between mentions. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 1595–1604, Melbourne, Australia. Association for Computational Linguistics.

Gina-Anne Levow. 2006. The third international Chinese language processing bakeoff: Word segmentation and named entity recognition. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 108–117, Sydney, Australia. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 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*, pages 7871–7880, Online. Association for Computational Linguistics.

Fei Li, ZhiChao Lin, Meishan Zhang, and Donghong Ji. 2021. A span-based model for joint overlapped and discontinuous named entity recognition. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, pages 4814–4828, Online. Association for Computational Linguistics.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 402–412, Baltimore, Maryland. Association for Computational Linguistics.

Xiaonan Li, Hang Yan, Xipeng Qiu, and Xuanjing Huang. 2020a. FLAT: Chinese NER using flat-lattice transformer. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6836–6842, Online. Association for Computational Linguistics.

Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2020b. A unified MRC framework for named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5849–5859, Online. Association for Computational Linguistics.

Hongyu Lin, Yaojie Lu, Xianpei Han, and Le Sun. 2019. Sequence-to-nuggets: Nested entity mention detection via anchor-region networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5182–5192, Florence, Italy. Association for Computational Linguistics.

Wei Lu and Dan Roth. 2015. Joint mention extraction and classification with mention hypergraphs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 857–867, Lisbon, Portugal. Association for Computational Linguistics.

Yi Luan, Dave Wadden, Luheng He, Amy Shah, Mari Ostendorf, and Hannaneh Hajishirzi. 2019. A general framework for information extraction using dynamic span graphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 3036–3046, Minneapolis, Minnesota. Association for Computational Linguistics.

Ying Luo and Hai Zhao. 2020. Bipartite flat-graph network for nested named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6408–6418, Online. Association for Computational Linguistics.

Yuxian Meng, Wei Wu, Fei Wang, Xiaoya Li, Ping Nie, Fan Yin, Muyu Li, Qinghong Han, Xiaofei Sun, and Jiwei Li. 2019. Glyce: Glyph-vectors for chinese character representations. In *Advances in Neural Information Processing Systems*. Curran Associates.

Xue Mengge, Bowen Yu, Zhenyu Zhang, Tingwen Liu, Yue Zhang, and Bin Wang. 2020. Coarse-to-Fine Pre-training for Named Entity Recognition. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6345–6354, Online. Association for Computational Linguistics.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1105–1116, Berlin, Germany. Association for Computational Linguistics.

Aldrian Obaja Muis and Wei Lu. 2017. Labeling gaps between words: Recognizing overlapping mentions with mention separators. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2608–2618, Copenhagen, Denmark. Association for Computational Linguistics.

Tomoko Ohta, Yuka Tateisi, and Jin-Dong Kim. 2002. The genia corpus: An annotated research abstract corpus in molecular biology domain. In *Proceedings of the Second International Conference on Human Language Technology Research*, page 82–86, San Francisco, USA. Morgan Kaufmann Publishers Inc.

Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Jie Ma, Alessandro Achille, Rishita Anubhai, Cicero Nogueira dos Santos, Bing Xiang, and Stefano Soatto. 2021. Structured prediction as translation between augmented natural languages. In *9th International Conference on Learning Representations, ICLR 2021*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, Sofia, Bulgaria. Association for Computational Linguistics.

Nicky Ringland, Xiang Dai, Ben Hachey, Sarvnaz Karimi, Cecile Paris, and James R. Curran. 2019. NNE: A dataset for nested named entity recognition in English newswire. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5176–5181, Florence, Italy. Association for Computational Linguistics.

Yongliang Shen, Xinyin Ma, Zeqi Tan, Shuai Zhang, Wen Wang, and Weiming Lu. 2021a. 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*, pages 2782–2794, Online. Association for Computational Linguistics.

Yongliang Shen, Xinyin Ma, Yechun Tang, and Weiming Lu. 2021b. A trigger-sense memory flow framework for joint entity and relation extraction. In *Proceedings of the Web Conference 2021*, WWW '21, page 1704–1715, New York, NY, USA. ACM.

Mohammad Golam Sohrab and Makoto Miwa. 2018. Deep exhaustive model for nested named entity recognition. In *Proceedings of the 2018 Conference*

on Empirical Methods in Natural Language Processing, pages 2843–2849, Brussels, Belgium. Association for Computational Linguistics.

Jana Straková, Milan Straka, and Jan Hajic. 2019. Neural architectures for nested NER through linearization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5326–5331, Florence, Italy. Association for Computational Linguistics.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. Ernie 2.0: A continual pre-training framework for language understanding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8968–8975.

Chuanqi Tan, Wei Qiu, Mosha Chen, Rui Wang, and Fei Huang. 2020. Boundary enhanced neural span classification for nested named entity recognition. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):9016–9023.

Zeqi Tan, Yongliang Shen, Shuai Zhang, Weiming Lu, and Yueting Zhuang. 2021. A sequence-to-set network for nested named entity recognition. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence, IJCAI-21*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Christopher Walker, Stephanie Strassel, and Kazuaki Maeda. 2006. Ace 2005 multilingual training corpus. linguistic. In *Linguistic Data Consortium, Philadelphia 57*.

Bailin Wang and Wei Lu. 2018. Neural segmental hypergraphs for overlapping mention recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 204–214, Brussels, Belgium. Association for Computational Linguistics.

Jue Wang, Lidan Shou, Ke Chen, and Gang Chen. 2020a. Pyramid: A layered model for nested named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5918–5928, Online. Association for Computational Linguistics.

Wen Wang, Yang Cao, Jing Zhang, and Dacheng Tao. 2022a. Fp-detr: Detection transformer advanced by fully pre-training. In *10th International Conference on Learning Representations, ICLR 2022*.

Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. 2021. Improving named entity recognition by external context retrieving and cooperative learning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 1800–1812, Online. Association for Computational Linguistics.

Xinyu Wang, Yongliang Shen, Jiong Cai, Tao Wang, Xiaobin Wang, Pengjun Xie, Fei Huang, Weiming Lu, Yueting Zhuang, Kewei Tu, Wei Lu, and Yong Jiang. 2022b. DAMO-NLP at SemEval-2022 Task 11: A Knowledge-based System for Multilingual Named Entity Recognition.

Yu Wang, Yun Li, Hanghang Tong, and Ziye Zhu. 2020b. HIT: Nested named entity recognition via head-tail pair and token interaction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6027–6036, Online. Association for Computational Linguistics.

Lu Xu, Zhanming Jie, Wei Lu, and Lidong Bing. 2021. Better feature integration for named entity recognition. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3457–3469, Online. Association for Computational Linguistics.

Hang Yan, Tao Gui, Junqi Dai, Qipeng Guo, Zheng Zhang, and Xipeng Qiu. 2021. A unified generative framework for various NER subtasks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, pages 5808–5822, Online. Association for Computational Linguistics.

Songlin Yang and Kewei Tu. 2022. Bottom-up constituency parsing and nested named entity recognition with pointer networks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020. Named entity recognition as dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6470–6476, Online. Association for Computational Linguistics.

Changmeng Zheng, Yi Cai, Jingyun Xu, Ho-fung Leung, and Guandong Xu. 2019. A boundary-aware neural model for nested named entity recognition. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, pages 357–366, Hong Kong, China. Association for Computational Linguistics.

Hengyi Zheng, Bin Qin, and Ming Xu. 2021. Chinese medical named entity recognition using crf-mt-adapt and ner-mrc. In *2th International Conference on Computing and Data Science*, pages 362–365.

A Datasets

GENIA (Ohta et al., 2002) is an English biology nested named entity dataset and contains 5 entity types, including `DNA`, `RNA`, `protein`, `cell line`, and `cell type` categories. Follow Yu et al. (2020), we use 90%/10% train/test split and evaluate the model on the last epoch.

ACE04 and ACE05 (Doddington et al., 2004; Walker et al., 2006) are two English nested datasets, each of them contains 7 entity categories. We follow the same setup as previous work Katiyar and Cardie (2018); Lin et al. (2019).

KBP17 (Ji et al., 2017) has 5 entity categories, including `GPE`, `ORG`, `PER`, `LOC`, and `FAC`. We follow Lin et al. (2019) to split all documents into 866/20/167 documents for train/dev/test set.

NNE (Ringland et al., 2019) is a English nested NER dataset with 114 fine-grained entity types. Follow Wang et al. (2020a), we keep the original dataset split and pre-processing.

FewNERD (Ding et al., 2021) is a large-scale English flat NER dataset with 66 fine-grained entity types. Follow Ding et al. (2021), we adopt a standard supervised setting.

CoNLL03 (Tjong Kim Sang and De Meulder, 2003) is an English dataset with 4 types of named entities: `LOC`, `ORG`, `PER` and `MISC`. Follow Yan et al. (2021); Yu et al. (2020), we train our model on the train and development sets.

OntoNotes (Pradhan et al., 2013) is an English dataset with 18 types of named entity, consisting of 11 types and 7 values. We use the same train, development, test splits as Li et al. (2020b).

Chinese MSRA (Levow, 2006) is a Chinese dataset with 3 named entity types, including `ORG`, `PER`, `LOC`. We keep the original dataset split and pre-processing.

In Table 6 and Table 7, we report the number of sentences, the number of sentences containing nested entities, the average sentence length, the total number of entities, the number of nested entities, the nesting ratio, the maximum and the average number of entities in a sentence on all datasets.

B Implementation Details

In default setting, we set the number of instance queries $M = 60$, and the total assignable quantity $Q = M \times 0.75 = 45$. To ensure that the assignable quantities of different entities are balanced, we randomly divide Q to different entities and adjust each division to be larger than Q/G, where G is the number of the ground-truth entities. When the number of entities is more than the total assignable quantity, we specify $Q = G$. We have also tried other configurations that will be discussed in Appendix D.3. We set L word-level transformer layers after BERT and set auxiliary losses in each layer. In the default setting L equals 5. We compare the effect of different auxiliary layers on the model performance, which will be discussed in Appendix D.1. Since the instance queries are randomly initialized and do not have query semantics at the initial stage of training, we first fix the parameters of BERT and train the model for 5 epochs, allowing the instance queries to initially learn the query semantics. When decoding entities, we filter out the predictions with localization probability and classification probability less than the threshold 0.6 and 0.8, respectively.

C Baselines

We compare PIQN with the following baselines:

- **ARN** (Lin et al., 2019) designs a sequence-to-nuggets architecture for nested mention detection, which first identifies anchor words and then recognizes the mention boundaries.
- **HIT** (Wang et al., 2020b) designs a head-tail detector and a token interaction tagger, which can leverage the head-tail pair and token interaction to express the nested structure.
- **Pyramid** (Wang et al., 2020a) presents a layered neural model for nested entity recognition, consisting of a stack of inter-connected layers.
- **Biaffine** (Yu et al., 2020) formulates NER as a structured prediction task and adopts a dependency parsing approach for NER.
- **BiFlaG** (Luo and Zhao, 2020) designs a bipartite flat-graph network with two subgraph modules for outermost and inner entities.
- **BERT-MRC** (Li et al., 2020b) formulates the NER task as a question answering task. They construct type-specific queries using semantic prior information for entity categories.

	ACE04			ACE05			KBP17			GENIA		NNE		
	Train	Dev	Test	Train	Dev	Test	Train	Dev	Test	Train	Test	Train	Dev	Test
#S	6200	745	812	7194	969	1047	10546	545	4267	16692	1854	43457	1989	3762
#NS	2712	294	388	2691	338	320	2809	182	1223	3522	446	28606	1292	2489
#E	22204	2514	3035	24441	3200	2993	31236	1879	12601	50509	5506	248136	10463	21196
#NE	10149	1092	1417	9389	1112	1118	8773	605	3707	9064	1199	206618	8487	17670
NR	45.71	46.69	45.61	38.41	34.75	37.35	28.09	32.20	29.42	17.95	21.78	83.27	81.11	83.36
AL	22.50	23.02	23.05	19.21	18.93	17.2	19.62	20.61	19.26	25.35	25.99	23.84	24.20	23.80
#ME	28	22	20	27	23	17	58	15	21	25	14	149	58	64
#AE	3.58	3.37	3.73	3.39	3.30	2.86	2.96	3.45	2.95	3.03	2.97	5.71	5.26	5.63

Table 6: Statistics of the ***nested*** datasets used in the experiments. #S: the number of sentences, #NS: the number of sentences containing nested entities, #E: the total number of entities, #NE: the number of nested entities, NR: the nesting ratio (%), AL: the average sentence length, #ME: the maximum number of entities in a sentence, #AE: the average number of entities in a sentence

	CoNLL03			OntoNotes			FewNERD			Chinese MSRA		
	Train	Dev	Test	Train	Dev	Test	Train	Dev	Test	Train	Dev	Test
#S	14041	3250	3453	49706	13900	10348	131965	18824	37648	41728	4636	4365
#E	23499	5942	5648	128738	20354	12586	340247	48770	96902	70446	4257	6181
AL	14.50	15.80	13.45	24.94	20.11	19.74	24.49	24.61	24.47	46.87	46.17	39.54
#ME	20	20	31	32	71	21	50	35	49	125	18	461
#AE	1.67	1.83	1.64	2.59	1.46	1.22	2.58	2.59	2.57	1.69	0.92	1.42

Table 7: Statistics of the ***flat*** datasets used in the experiments. #S: the number of sentences, #E: the total number of entities, AL: the average sentence length, #ME: the maximum number of entities in a sentence, #AE: the average number of entities in a sentence

- **BARTNER** (Yan et al., 2021) formulates NER as an entity span sequence generation problem and uses a unified Seq2Seq model with the pointer mechanism to tackle flat, nested, and discontinuous NER tasks.

- **Seq2Set** (Tan et al., 2021) formulates NER as an entity set prediction task. Different from Straková et al. (2019), they utilize a non-autoregressive decoder to predict entity set.

- **Locate&Label** (Shen et al., 2021a) treats NER as a joint task of boundary regression and span classification and proposed a two-stage identifier of locating entities first and labeling them later.

For a fair comparison, we did not compare with Sun et al. (2020); Li et al. (2020a); Meng et al. (2019) on Chinese MSRA because they either used glyphs or an external lexicon or a larger pre-trained language model. In addition, some works (Wang et al., 2021, 2022b) used search engines to retrieve input-related contexts to introduce external information, and we did not compare with them as well.

D Analysis

D.1 Analysis of Auxiliary Loss

Many works (Al-Rfou et al., 2019; Carion et al., 2020) have demonstrated that the auxiliary loss in the middle layer introduces supervised signals in advance and can improve model performance. We compared the effect of the different number of auxiliary-loss layers on the model performance (F1-score on ACE04). Overall, the model performs better as the number of auxiliary-loss layers increases. The model achieves the best results when the number of layers equals 5.

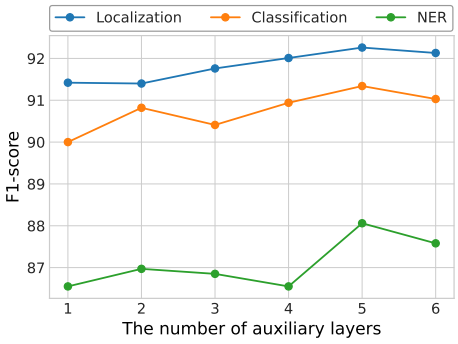


Figure 5: Analysis of Auxiliary Loss

D.2 Analysis of Two Subtasks

We compare the model performance on entity localization and entity classification subtasks on the ACE04 dataset, as shown in Table 8. Compared with the previous state-of-the-art models (Tan et al., 2021; Shen et al., 2021a), our model achieves better performance on both entity localization and entity classification subtasks. This illustrates that the instance queries can automatically learn their query semantics about location and type of entities, which is consistent with our analysis in 5.3.

Model	Localization		
	Pr.	Rec.	F1
Tan et al. (2021)	92.75	90.24	91.48
Shen et al. (2021a)	92.28	90.97	91.62
PIQN	92.56	**91.89**	**92.23**
Model	Classification		
	Pr.	Rec.	F1
Tan et al. (2021)	95.36	86.03	90.46
Shen et al. (2021a)	95.40	86.75	90.87
PIQN	**95.59**	**87.81**	**91.53**

Table 8: Localization and Classification Performance on ACE04

D.3 Analysis of Label Assignment

(M, Q)	Loc. F1	Cls. F1	Pr.	Rec.	F1
(60, 15)	91.05	90.15	87.57	85.67	86.61
(60, 30)	91.76	90.37	88.23	86.16	87.18
(60, 45)	**92.23**	**91.53**	**88.48**	**87.81**	**88.14**
(60, 50)	92.01	90.81	87.38	87.12	87.25
(30, 15)	91.26	89.66	88.61	84.88	86.70
(60, 30)	91.76	90.37	88.23	86.16	87.18
(90, 45)	**91.88**	**90.56**	**88.23**	**86.46**	**87.34**
(120, 60)	91.75	90.45	87.19	86.56	86.87

Table 9: Analysis on Dynamic Label Assignment for different combinations of the number M of instance queries and the total assignable quantity Q of labels.

We analyze the impact of dynamic label assignment on model performance for different combinations of the number M of instance queries and the total assignable quantity Q of labels. From Table 9, we observe that (1) there is a tradeoff between M and Q, and the model achieves the best performance with a ratio of 4:3. With this setting, the ratio of positive to negative instances of instance queries is 3:1. (2) The number of instance queries and the total assignable quantity is not as large as possible, and an excessive number may degrade the model performance. In our experiments $(M, Q) = (60, 45)$ is the best combination.