NAG-NER: a Unified Non-Autoregressive Generation Framework for Various NER Tasks

Xinpeng Zhang[1], **Ming Tan**[2*], **Jingfan Zhang**[3*], **Wei Zhu**[4*†]
[1]NetEase Youdao
[2]Southern University of Science and Technology
[3] University of Ottawa
[4]East China Normal University

Abstract

Recently, the recognition of flat, nested, and discontinuous entities by a unified generative model framework has received increasing attention both in the research field and industry. However, the current generative NER methods force the entities to be generated in a predefined order, suffering from error propagation and inefficient decoding. In this work, we propose a unified non-autoregressive generation (NAG) framework for general NER tasks, referred to as NAG-NER. First, we propose to generate entities as a set instead of a sequence, avoiding error propagation. Second, we propose incorporating NAG in NER tasks for efficient decoding by treating each entity as a target sequence. Third, to enhance the generation performances of the NAG decoder, we employ the NAG encoder to detect potential entity mentions. Extensive experiments show that our NAG-NER model outperforms the state-of-the-art generative NER models on three benchmark NER datasets of different types and two of our proprietary NER tasks.

1 Introduction

Named entity recognition (NER) is a fundamental task in the field of information extraction. It is the basic task for many natural language processing applications like dialogue systems, document analysis, and search engines. Currently, NER tasks can be divided into three subtasks (Yan et al., 2021), i.e., flat NER, nested NER, and discontinuous NER, as illustrated in Figure 1. Recently, researchers have grown interested in tackling the three subtasks via a unified model architecture, which we refer to as general NER models (Li et al., 2020; Dai et al., 2020; Yan et al., 2021). Existing literature for general NER models fall into the following three categories: (1) span-based models (Yu et al., 2020a; Bekoulis et al., 2018); (2) models based on carefully designed data structures like hyper-graphs and shift-reduce parsers (Dai et al., 2020; Wang et al., 2021b); (3) sequence-to-sequence (seq2seq) models (Yan et al., 2021; Zhang et al., 2022).

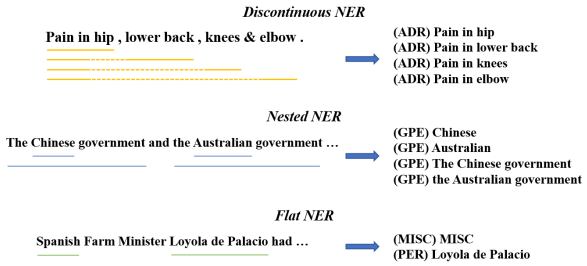

Figure 1: Examples of the discontinuous / nested / flat NER.

Among the three branches of literature, the seq2seq models (Yan et al., 2021; Fei et al., 2021) have achieved SOTA performances. However, they organize target entities into a single sequence according to a predetermined order. This setting is against the intuition that the target entities are essentially an unordered set and results in an incorrect bias (entity-order confounder) to the model (Zhang et al., 2022). In addition, sequentially generating target entities suffers from two disadvantages: (1) low inference speed due to autoregressive decoding; (2) Error propagation, i.e., errors generated by the previous steps could misguide the current and future generation steps.

In this paper, we propose a non-autoregressive generation (NAG) framework for named entity recognition, *NAG-NER* (as depicted in Figure 2). Given an input sentence, the framework first encodes the sentence and detects where and how many entities will start at each token of the input sentence. Then, it asks the decoder to generate the set of targeted entities accordingly. We conducted extensive experiments on three benchmark datasets (CADEC, ACE2004, CoNLL03) and two proprietary datasets we developed (referred to as CME

*Equal contribution.
†Corresponding author: michaelwzhu91@gmail.com

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 5: Industry Track, pages 676–686

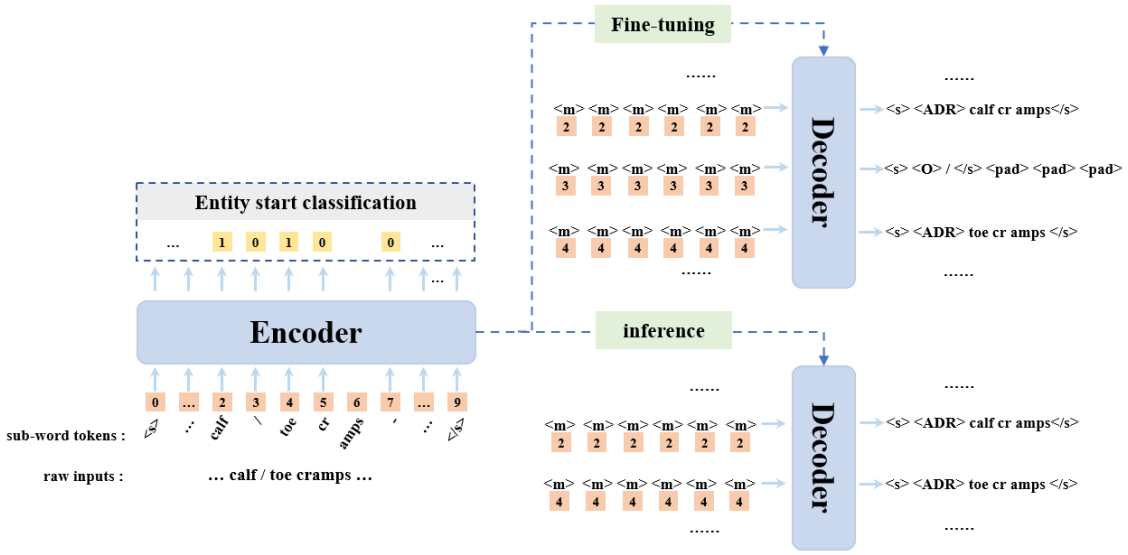

Figure 2: The overall framework of our NAG-NER framework. **Left**: the encoder is tasked for obtaining high-quality context representations for the input text, and entity start classification. **Upper right**: During training, the decoder will take a sequence of $\langle m \rangle$ tokens, and generate the corresponding entity type label and the whole entity mention. **Lower right**: During inference, the encoder first propose the entity starting words and #entities starting from these words, and then the decoder will generate corresponding entity results.

and QER). The results validate that our method can perform comparably to or outperform the previous generative NER models while achieving significant speedups.

To summarize, our main contributions include the following:

- We propose NAG-NER, a novel non-autoregressive entity generation framework for general NER tasks. Distinct from the seq2seq models, it avoids the entity-order confounder and error propagation by generating multiple entity sequences simultaneously with a pre-trained non-autoregressive generation model.
- Experimental results show that our model achieves SOTA performances while being efficient.

2 Related Work

Due to limited length, we include the related works of NAG in Appendix A. We also include a preliminary introduction of NAG in Appendix B.

2.1 Generative NER models

Generative models are investigated to solve different types of NER tasks in a unified model framework. Straková et al. (2019) propose to transform the BILOU labels (Ratinov and Roth, 2009) of source tokens into a label sequence via heuristic rules. Athiwaratkun et al. (2020) propose an augmented natural language output format for flat NER tasks, where the type tags of words are placed along with the words to form a sentence-like target sequence. Tan et al. (2021) proposes to generate entities as a set. However, this model directly generates entity position and type information via a set generation framework, failing to employ the text generation capabilities of pre-trained generative models. Yan et al. (2021) combine pre-trained BART with a delicately-designed copying mechanism and achieve promising performance on a wide range of NER benchmarks. Fei et al. (2021) train an LSTM from scratch to generate the target sequence and devise a novel memory-augmented pointer mechanism to enhance the interactions between the current pointer and the prior recognized entity mentions. Lu et al. (2022) transforms different information extraction task into a structured extraction language and solve general information extraction tasks with a unified text-to-structure generation framework. Zhang et al. (2022) point out two kinds of incorrect bias (pre-context confounder, entity-order confounder) in the generative NER models and propose two data augmentation methods to address these biases, but this model still has to generate all the entities sequentially.

Our NAG-NER framework contributes to the literature by (a) we generate all the entities of a given input sentence in parallel via a NAG model, resulting in significant speedup and avoiding the pre-context confounder; (b) our framework bypasses the entity-order confounder since it generates a set of entities in a single forward pass.

3 Methods

In detail, we formally introduce our framework (Figure 2). We uniformly formulate the task of recognizing flat, nested, and discontinuous entities as NAG-based entity sequence generation problems. We will take a pre-trained NAG model consisting of an encoder and a decoder as the model backbone. Denote the tokenized (Sennrich et al., 2016) input sentence with length L as $S = [w_0, ..., w_{L-1}]$. And the target output is the set of entity sequences $\{(\text{ES}_i, T_i)\}_{i=1}^{M}$, where M is the number of entities, ES_i is the i-th entity consisting of l_i tokens, and T_i is the type label of entity i.

3.1 Encoder

The pre-trained NAG encoder will encode the input sequence S and output the contextualized representations:

$$H = \textbf{Encoder}(S), \tag{1}$$

where $H \in \mathcal{R}^{L \times d}$ and d is the hidden size of the NAG model. Since the transformer-based NAG model will provide the contextualized encoding of tokens, we use the representations of each word's first subword token as the vector representation of this word:

$$v_j = H[\text{start}_j], \tag{2}$$

where v_j is the representation of the j-th word in the original input sequence, start_j is the index of the first subword token of word j in sequence S.

3.2 Entity start classification

To motivate the encoder to gain a deep understanding of the input sentence and provide information for the decoder, we ask the encoder to detect where an entity will start and how many entities will start at such a position.[1] This task is formulated as a multi-class classification task on each word j predicting the number of entities ne_j that starts at word j:

$$\mathbf{p}(\text{ne}_j) = \text{Softmax}(v_j W_{\text{ne}} + b_{ne}), \tag{3}$$

where $\mathbf{p}(\text{ne}_j) \in \mathbf{R}^{O_{max}+1}$, O_{max} is the maximum number of entities starting at the same word. Note that $\text{ne}_j = 0$ means that no entity will start at word j. This task is optimized with cross-entropy loss $\mathcal{L}_{\text{ne}}$.

3.3 Entity generation

As shown in Figure 2, we ask the NAG decoder to generate entity information for each word j of the input. The non-autoregressive generation (NAG) model is proposed (Gu et al., 2018) to speedup autoregressive generation, which removes the order dependency between target tokens Y and can generate tokens of the target sentence simultaneously given input X:

$$\mathbf{P}_{NAG}(Y|X;\Theta) = \prod_{t=1}^{l} \mathbf{P}(y_t|X;\Theta), \tag{4}$$

where l denotes the length of the target sentence. The decoder of the NAG model needs to know the targeted length before generation. A common practice is to treat length prediction as a classification task, using the information from the encoder's output to make predictions. However, following Qi et al. (2020), we will discard this length prediction task by using a unified length for the output sequence and use the first generated end-of-sentence token $\langle /s \rangle$ as the ending signal for the generated sequence.

Denote the maximum entity length (in subword level) as $l_{max} \in \mathbf{Z}_+$. During training, we sort (in ascending order) the ne_j entities by their spanning length, that is, the length of the span that envelops this entity.[2] For each word j and each $n = 0, 1, 2, ..., \text{ne}_j - 1$, we would like the decoder to generate entity $(\text{ES}_{j,n}, \text{T}_{j,n})$. In the negative cases where no named entities are starting from word j, $\text{ES}_{j,n}$ will be the subtokens of word j, and $\text{T}_{j,n}$ will be the non-entity tag $\langle \text{O} \rangle$. Thus, the input and target output of the decoder are:

- Decoder input: a sequence of length $l_{max} + 3$ consisting of only the mask token $\langle m \rangle$ are fed into the decoder;
- Targeted decoder output: the target output sequence is in the form of

$$\langle s \rangle \ \text{ET}_{j,n} \ \text{ES}_{j,n} \ \langle /s \rangle, \tag{5}$$

[1] For discontinuous and nested NER tasks, multiple entities could share the same starting words, as shown in Figure 1.

[2] This is reflected on Figure 1, where the spanning length of "Pain in hip" is smaller than that of "Pain in knees".

where $\text{ES}_{j,n}$ is the subtoken sequences of this entity, and $\text{ET}_{j,n}$ is the token added to the NAG vocabulary that represents the entity class tag $\text{T}_{j,n}$. The non-entity tag O corresponds to the special token $\langle O \rangle$.

Note that since entities are of different lengths, we will pad the target output sequence with the padding token $\langle \text{pad} \rangle$ to length $l_{max}+3$ if necessary.

The model does not need to generate entity sequences on every word during inference. We can take advantage of the entity start classification module and decide which words are likely to be the starting words of named entities. Formally, with the threshold τ_s $(0 < \tau_s < 1)$ for entity start classification,

$$S_{start} = \{j \mid \mathbf{p}(\text{ne}_j = 0) < \tau_s\}, \tag{6}$$

where S_{start} is the collection of indexes of detected entity starting words, and the number of entities is obtained by $\text{ne}_j \leftarrow \arg\max_{\text{ne}_j} \mathbf{p}(\text{ne}_j)$. Then we will generate entity sequences by feeding the decoder a sequence of length $l_{max}+3$ consisting of only the mask token $\langle m \rangle$ for each $j \in S_{start}$ and each $n < \mathbf{ne}_j$. After the decoder generates an output sequence, the entity tag token $\text{ET}_{j,n}$ is obtained as the second generated token, and the token sequence from the third position till the first $\langle /s \rangle$ token is the generated entity token sequence. If there is no $\langle /s \rangle$ token in the output sequence, all the output tokens starting from the third position will be considered the generated entity token sequence.

Given a decoder input and targeted output sequences, we can calculate the generation loss $\mathcal{L}_g$ of NAG, which is the average cross-entropy loss on each token according to Equation 4. We will discard the losses from $\langle \text{pad} \rangle$ tokens.

3.4 Positional embeddings for entity generation

Note that the decoder receives input sequences consisting of only mask tokens $\langle m \rangle$, and it does not know where the entity starts and which contexts it should pay more attention to during generation. In addition, we should also inform the decoder about the number of entities starting from the same word so that the decoder can generate entities of different spanning lengths instead of generating the same entity. Thus, we introduce two positional embedding to the decoder's embedding layer:

- **word start position embedding (WSPE)**: as depicted in Figure 2, all the mask tokens of the decoder input share the same word start position index, that is, start_j, the index of the first subtoken of word j. Furthermore, the word start position embedding vector is obtained by looking up the positional embedding layer of the encoder.
- **number-of-entities position embedding (NEPE)**: for each $n < \mathbf{ne}_j$, n represents the n-th entity, and the n-th shortest entity starting from word j. We map n to a randomly initialized learnable embedding vector NEPE_n. This positional embedding is also shared by all the tokens of the decoder input.

These two positional embeddings will be added to the decoder's original embedding layer.

3.5 Overall fine-tuning objective

During fine-tuning of NAG, the whole framework of NAG-NER is optimized end-to-end, with the total losses of entity start classification and entity sequence generation:

$$\mathcal{L} = \mathcal{L}_g + \mathcal{L}_{\text{ne}}. \tag{7}$$

4 Experiments

4.1 Evaluation datasets and metrics

To show that our proposed method can be used in various NER subtasks, we conducted experiments on three English open-sourced benchmark datasets (CADEC, ACE2004, CoNLL03) and two Chinese proprietary tasks (CME, QER). CoNLL03 and QER are flat NER tasks, CADEC is a discontinuous NER task, and ACE2004 contains nested entities. CME is a complex task containing both discontinuous and nested entities. We include introductions and statistics of the datasets in Appendix C.

For evaluation, strict evaluation metrics are applied, where an entity is confirmed correct only if all of its words and its type label are recognized correctly. Precision (P), Recall (R), and Micro F1 score (F1) are reported in the results.

4.2 Implementation Details

We employ the BANG model (Qi et al., 2020) as the backbone for English tasks while we pre-train a NAG model in Chinese based on the codebase of BANG on our corpus containing 120 million documents in Chinese. For fine-tuning on each task, the

special tokens corresponding to the entity type labels (including the non-entity label ⟨O⟩) are added to the vocabulary, and their embedding vectors are randomly initialized. For optimization, we use the AdamW optimizer (Loshchilov and Hutter, 2018) with a linear learning rate schedule and 6% of the optimization steps as warm-up steps. After each epoch, we evaluate the fine-tuned model on the development set and save the model checkpoints. After fine-tuning ends, the best checkpoint will be evaluated on the test set, and the test result will be reported. Details of hyper-parameter tuning and settings are included in Appendix D. We report the average test performance on five random seeds.

4.3 Compared Methods

We mainly compare our model with SOTA generative NER models listed in Section 2. We also compare our method with SOTA discriminative NER models. See Appendix E for an introduction to them.

For a fair comparison, since our NAG model is in the base size, we run the baseline models with BERT-base (Devlin et al., 2019) (12 encoder layers) or BART-base (Lewis et al., 2019) (6 encoder layers and 6 decoder layers).[3] Lu et al. (2022) is run with the implementation of PaddlePaddle[4]. All the baselines are run with their open-sourced codes with their suggested hyper-parameters.

4.4 Main results

Table 1 and Table 2 show the comparison between our model and other models in three benchmark datasets and two proprietary datasets.

Results on the open-sourced benchmark datasets Table 1 demonstrates that on the benchmark datasets, our method has clear advantages over the previous SOTA generative methods on complex discontinuous or nested NER tasks CADEC and ACE2004. On these tasks, our method also outperforms the models designated for specific tasks, like Wang et al. (2021b). On the flat NER tasks, although the previous generative models slightly underperform the discriminative models, our method can obtain results comparable to the strong discriminative models, demonstrating the broad applicability of our method.

[3]The Chinese version of BART-base is provided by https://huggingface.co/fnlp/bart-base-chinese.

[4]https://github.com/PaddlePaddle/PaddleNLP/tree/develop/model_zoo/uie

Model	CADEC F1	ACE2004 F1	CoNLL03 F1
Discriminative NER models			
Tang et al. (2018)	65.1	-	-
Dai et al. (2020) [ELMO]	68.7	-	-
Wang et al. (2021b) [BERT-base]	69.7	-	-
Yu et al. (2020a) [BERT-base]	-	85.6	92.4
Li et al. (2020) [BERT-base]	-	84.2	92.7
Xu et al. (2021) [BERT-base]	-	85.0	-
Shen et al. (2021) [BERT-base]	-	85.7	92.7
Akbik et al. (2019) [BERT-base]	-	-	92.8
Wang et al. (2021a) [BERT-base]	-	-	92.8
Set Generation NER models			
Tan et al. (2021) [BERT-base]	-	85.6	92.4
Generative NER models			
Straková et al. (2019) [BART-base]	-	84.3	92.4
Yan et al. (2021) [BART-base]	68.7	85.2	92.5
Fei et al. (2021) [BART-base]	70.6	-	-
Zhang et al. (2022) [BART-base]	70.8	85.2	92.7
Lu et al. (2022) [BERT-base]	-	85.3	92.3
NAG-NER (ours)	**71.3**	**85.9**	**92.8**

Table 1: Results on the three NER benchmark datasets. The results show that our NAG-NER method has clear advantages on complex NER tasks while performing comparably with the SOTA models on flat NER tasks.

Model	CME F1	QED F1
Discriminative NER models		
Yu et al. (2020a) [BERT-base]	88.9	-
Li et al. (2020) [BERT-base]	87.8	94.5
Shen et al. (2021) [BERT-base]	88.7	94.6
Akbik et al. (2019) [BERT-base]	-	94.5
Wang et al. (2021a) [BERT-base]	-	94.8
Set Generation NER models		
Tan et al. (2021) [BERT-base]	-	93.8
Generative NER models		
Straková et al. (2019) [BART-base]	86.4	93.9
Yan et al. (2021) [BART-base]	88.6	94.2
Zhang et al. (2022) [BART-base]	88.5	94.3
Lu et al. (2022) [BERT-base]	88.8	94.4
NAG-NER (ours)	**89.6**	**94.7**

Table 2: Results on the two proprietary datasets, CME and QER.

Results on the proprietary datasets The results on the our proprietary dataset (Table 2) lead to similar observations with Table 1. Our method outperforms the baseline methods on the complex task CME which contains both discontinuous and nested entities. In addition, the performance of our method on the flat NER task, QER, is also comparable to the strong discriminative baseline models.

4.5 Inference Efficiency

We compare the inference efficiency of our method with the SOTA seq2seq ner model Yan et al. (2021) and the SOTA set generation NER model Tan et al. (2021) on two tasks: ACE04 and

Methods	QPS	
	ACE2004	**CME**
Yan et al. (2021) [BART-base]	108 (1×)	11
Tan et al. (2021) [BERT-base]	227 (2.1×)	-
NAG-NER (ours)	205 (1.9×)	63 (5.7×)

Table 3: Comparison of efficiency for three models, using a NVIDIA RTX 3090 GPU. The results show that our method can effectively speed up inference for various NER tasks.

CME. We run each model repeatedly on a fixed batch of samples containing four sentences for a fair comparison. For ACE2004, the batch contains 86 tokens; for CME, the batch contains 892 tokens. The efficiency is measured on an NVIDIA RTX 3090 GPU. We report the average number of sentences processed per second (QPS) of each model in Table 3. As shown in Table 3, our method is significantly faster than the seq2seq NER model and only runs slightly slower than Tan et al. (2021). Note that our CME task has a much longer average sentence length and a larger number of entities per sentence. Thus the speedup effects of our NAG-NER method on CME are much more significant than on the ACE2004 task.

4.6 Ablation studies

We conduct an ablation study on ACE2004 and CME to verify the effectiveness of different components of NAG-NER. We consider three different variations of our whole NAG-NER model whose results are presented in Table 4:

- NAG-NER-1. In our main experiments (in Table 2), we utilize a Chinese BANG model to initialize our model for the CME and QER tasks. This BANG model is pre-trained on our Chinese corpus with 100 thousand steps under a batch size of 1024. We now substitute this pre-trained checkpoint with a less well-pre-trained one (at 20 thousand steps). NAG-NER-1 under-performs NAG-NER on the CME test set, demonstrating that the quality of the pre-trained NAG models can directly affect the results of our method.

- NAG-NER-2, which is to drop the entity start classification module. Thus, in this model, the decoder has to generate entity sequences on each word's starting token. After dropping this module, the F1 score drops slightly on both tasks, showing that this module is beneficial for filtering out noise and increasing the precision of the decoder's generation outputs.

- NAG-NER-3, which is to drop the WSPE and NEPE positional embeddings in Section 3.4. NAG-NER-3 can not obtain a reasonable performance, demonstrating the necessity of informing our decoder where the entity starts to generate the entity mentions correctly.

Methods	**ACE2004**	**CME**
	F1	F1
NAG-NER	**85.9**	**89.6**
NAG-NER-1	-	88.7
NAG-NER-2	85.3	88.9
NAG-NER-3	73.2	69.5

Table 4: Results of ablation study on the ACE2004 and CME tasks.

Dataset	**Test set**	Yan et al. (2021)	NAG-NER
ACE2004	All	85.2	85.9 (+0.7)
	Overlapping	83.2	84.7 (+1.5)
CME	All	88.6	89.6 (+1.0)
	Overlapping	83.4	85.8 (+2.4)

Table 5: Results of ablation study on the ACE2004 and CME tasks.

4.7 Error analysis

To further demonstrate the advantage of our method over Seq2Seq NER models, we now analyze how our model performs when dealing with overlapping entities. In Table 5, we report the F1 scores on the whole test set and the subset of overlapping entities for the ACE2004 and CME tasks. We can see that compared with Yan et al. (2021), our NAG-NER model significantly boosts the F1 score on the overlapping entities, showing that our method is effective in recognizing complex entities.

5 Conclusion

In this work, we propose NAG-NER, a unified generative model for various NER tasks based on non-autoregressive generation (NAG). In NAG-NER, different NER tasks are formulated as entity set generation tasks. We employ the NAG encoder to detect potential entity starts and the NAG decoder to efficiently decode entity sequences. Experiments on three benchmark NER tasks and two proprietary NER tasks demonstrate that our method can outperform baseline generative NER methods while achieving higher inference speed. We also conduct ablation studies to demonstrate the necessity of each module in our NAG-NER method.

Limitations

In this work, we develop a unified model framework that is applicable to different NER tasks. Through experiments, we show the effectiveness of our method on different NER tasks, both in English and Chinese. However, we recognize that our method is not tested on NER tasks where the input sequences are extremely long. In addition, our method is not tested on few-shot scenarios. We will investigate these issues in future work.

Ethics Statement

Our model is designated to recognize entities in input sequences. We use two groups of tasks. The three benchmark datasets CADEC, ACE2004, and CoNLL03 are widely studied in the literature, and our work does not introduce new ethical issues. Since the two proprietary datasets are all anonymized and only used for training models in our institution, no ethical concerns are included in our work.

References

A. Akbik, Tanja Bergmann, and Roland Vollgraf. 2019. Pooled contextualized embeddings for named entity recognition. In *North American Chapter of the Association for Computational Linguistics*.

Ben Athiwaratkun, Cícero Nogueira dos Santos, Jason Krone, and Bing Xiang. 2020. Augmented natural language for generative sequence labeling. In *Conference on Empirical Methods in Natural Language Processing*.

Yu Bao, Hao Zhou, Shujian Huang, Dongqi Wang, Lihua Qian, Xinyu Dai, Jiajun Chen, and Lei Li. 2022. Glat: Glancing at latent variables for parallel text generation. *ArXiv*, abs/2204.02030.

Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder. 2018. Joint entity recognition and relation extraction as a multi-head selection problem. *Expert Systems with Applications*, 114:34–45.

Xiang Dai, Sarvnaz Karimi, Ben Hachey, and Cecile Paris. 2020. An effective transition-based model for discontinuous NER. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5860–5870, Online. Association for Computational Linguistics.

Keqi Deng, Zehui Yang, Shinji Watanabe, Yosuke Higuchi, Gaofeng Cheng, and Pengyuan Zhang. 2022. Improving non-autoregressive end-to-end speech recognition with pre-trained acoustic and language models. *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8522–8526.

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: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Hao Fei, Dong-Hong Ji, Bobo Li, Yijiang Liu, Yafeng Ren, and Fei Li. 2021. Rethinking boundaries: End-to-end recognition of discontinuous mentions with pointer networks. In *AAAI Conference on Artificial Intelligence*.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. 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)*, pages 6112–6121, Hong Kong, China. Association for Computational Linguistics.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O.K. Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*.

Jiatao Gu and Xu Tan. 2022. Non-autoregressive sequence generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, ACL 2022 - Tutorial Abstracts, Dublin, Ireland, May 22-27, 2022*, pages 21–27. Association for Computational Linguistics.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *North American Chapter of the Association for Computational Linguistics*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Annual Meeting of the Association for Computational Linguistics*.

Junyi Li, Tianyi Tang, Wayne Xin Zhao, Jianyun Nie, and Ji rong Wen. 2022. Elmer: A non-autoregressive pre-trained language model for efficient and effective text generation. *ArXiv*, abs/2210.13304.

Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2020. 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.

Jindřich Libovický and Jindřich Helcl. 2018. End-to-end non-autoregressive neural machine translation with connectionist temporal classification. *ArXiv*, abs/1811.04719.

Ilya Loshchilov and Frank Hutter. 2018. Fixing weight decay regularization in adam.

Yaojie Lu, Qing Liu, Dai Dai, Xinyan Xiao, Hongyu Lin, Xianpei Han, Le Sun, and Hua Wu. 2022. Unified structure generation for universal information extraction. In *Annual Meeting of the Association for Computational Linguistics*.

Aldrian Obaja Muis and Wei Lu. 2017. Labeling gaps between words: Recognizing overlapping mentions with mention separators. In *Conference on Empirical Methods in Natural Language Processing*.

Weizhen Qi, Yeyun Gong, Jian Jiao, Yu Yan, Dayiheng Liu, Weizhu Chen, Kewen Tang, Houqiang Li, Jiusheng Chen, Ruofei Zhang, Ming Zhou, and Nan Duan. 2020. Bang: Bridging autoregressive and non-autoregressive generation with large scale pretraining. *ArXiv*, abs/2012.15525.

Lev-Arie Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Conference on Computational Natural Language Learning*.

Chitwan Saharia, William Chan, Saurabh Saxena, and Mohammad Norouzi. 2020. Non-autoregressive machine translation with latent alignments. In *Conference on Empirical Methods in Natural Language Processing*.

Erik Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *ArXiv*, cs.CL/0306050.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

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

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. In *International Conference on Machine Learning*.

Matteo Stefanini, Marcella Cornia, Lorenzo Baraldi, Silvia Cascianelli, Giuseppe Fiameni, and Rita Cucchiara. 2021. From show to tell: A survey on deep learning-based image captioning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45:539–559.

Jana Straková, Milan Straka, and Jan Hajic. 2019. Neural architectures for nested ner through linearization. In *Annual Meeting of the Association for Computational Linguistics*.

Shuo Sun, Hongxu Hou, Nier Wu, Ziyue Guo, and Chaowei Zhang. 2020. Multi-reward based reinforcement learning for neural machine translation. In *CCL*.

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

Buzhou Tang, Jianglu Hu, Xiaolong Wang, and Qingcai Chen. 2018. Recognizing continuous and discontinuous adverse drug reaction mentions from social media using lstm-crf. *Wirel. Commun. Mob. Comput.*, 2018.

Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. 2021a. Improving named entity recognition by external context retrieving and cooperative learning. In *Annual Meeting of the Association for Computational Linguistics*.

Yucheng Wang, Bowen Yu, Hongsong Zhu, Tingwen Liu, Nan Yu, and Limin Sun. 2021b. Discontinuous named entity recognition as maximal clique discovery. In *Annual Meeting of the Association for Computational Linguistics*.

Yongxiu Xu, Heyan Huang, Chong Feng, and Yue Hu. 2021. A supervised multi-head self-attention network for nested named entity recognition. In *AAAI Conference on Artificial Intelligence*.

Hang Yan, Tao Gui, Junqi Dai, Qipeng Guo, Zheng Zhang, and Xipeng Qiu. 2021. A unified generative framework for various ner subtasks. In *Annual Meeting of the Association for Computational Linguistics*.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020a. 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.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020b. Named entity recognition as dependency parsing. In *Annual Meeting of the Association for Computational Linguistics*.

Chengchang Zeng, Shaobo Li, Qin Li, Jie Hu, and Jianjun Hu. 2020. A survey on machine reading comprehension: Tasks, evaluation metrics, and benchmark datasets. *ArXiv*, abs/2006.11880.

Shuai Zhang, Yongliang Shen, Zeqi Tan, Yiquan Wu, and Weiming Lu. 2022. De-bias for generative extraction in unified ner task. In *Annual Meeting of the Association for Computational Linguistics*.

A Appendix of related work

A.1 Non-autoregressive Generation Models

Due to its advantages in efficiency, there is a wide range of studies for NAG models (Gu et al., 2018; Ghazvininejad et al., 2019; Qi et al., 2020). Gu et al. (2018) is the first to propose NAG paradigm to reduce the inference latency of text generation. NAG is widely studied in machine translation. Ghazvininejad et al. (2019) masks and predicts a fraction of tokens that the model is least confident about. Saharia et al. (2020) and Libovický and Libovický and Helcl (2018) use connectionist temporal classification to perform latent alignment in NAR models. Bao et al. (2022) employs the discrete latent variables to capture word categorical information and invoke an advanced curriculum learning technique, alleviating the multi-modality problem of NAG in machine translation tasks. Recently, several groups aim to apply NAG to a wider range of tasks. Qi et al. (2020) designs and pre-trains a monoligual Transformer model with multiple attention streams that can be used both as an AG model and a NAG model. They apply their pre-trained models on summarization, machine reading comprehension and dialogue response generation, and show that NAG models can achieve competitive performance with around 15 times speedup. Li et al. (2022) develops an early exiting based strategies for monoligual NAG pre-training.

Our work complements the literature by: (a) we successfully apply NAG models in named entity recognition tasks; (b) we propose a span set generation task for pre-training a NAG model which is more suitable for downstream NER tasks.

B Appendix: Preliminaries on NAG

B.1 Autoregressive generation

The autoregressive generation (AG) models achieve the state-of-the-art performance on a wide range of text generation tasks like machine translation (Song et al., 2019; Sun et al., 2020), summarization (Lewis et al., 2019), image captioning (Stefanini et al., 2021). We now use machine translation to introduce the AG method. Given a source sentence $X = (x_1, x_2, ..., x_n)$ and the target sentence $Y = (y_1, y_2, ..., y_m)$, an AG model with parameters Θ decomposes the target distribution of translations according to the chain rule:

$$\mathbf{P}_{AG}(Y|X;\Theta) = \prod_{t=1}^{m} \mathbf{P}(y_t|y_{<t}, X;\Theta), \quad (8)$$

where $y_{<t}$ denotes generated previous tokens before the t-th position. During the training process, the AG model is usually trained via the teacher-forcing strategy that uses ground truth target tokens as previously decoded tokens so that the output of the decoder can be computed in parallel. During inference, the AG model still needs to generate translations one-by-one from left to right until the token $\langle /s \rangle$ that represents the end of sentence. Although AG models achieve SOTA performances on text generation tasks, its autoregressive decoding method dramatically reduces the decoding speed and becomes the main bottleneck of its efficiency. In addition, some literature argues that autoregressive decoding is prone to error propagation Gu and Tan (2022).

B.2 Non-autoregressive generation

To improve the inference speed of AG models, the non-autoregressive generation (NAG) model is proposed (Gu et al., 2018), which removes the order dependency between target tokens and can generate tokens of the target sentence simultaneously:

$$\mathbf{P}_{NAG}(Y|X;\Theta) = \prod_{t=1}^{m} \mathbf{P}(y_t|X;\Theta), \quad (9)$$

where m denotes the length of the target sentence. Generally, NAG models need to have the ability to predict the length because the entire sequence needs to be generated in parallel. A common practice is to treat it as a classification task, using the information from the encoder's output to make predictions. Qi et al. (2020) discard the length prediction module and use the first generated end-of-sentence token $\langle /s \rangle$ as the ending signal for the generated sequence.

Because NAG is only conditioned on source-side information, but AG can obtain the strong target-side context information provided by the previously generated target tokens, NAG models generally have a performance gap compared to AG models. NAG is first studied in machine translation and

Statistics	*CADEC*			*ACE04*			*CoNLL03*		
	Train	Dev	Test	Train	Dev	Test			
# Sentences	6077	760	759	6200	745	812	14041	3250	3453
Avg sent. length	14.1	14.2	14.2	22.5	23.0	23.0	13.7	13.5	13.6
# Entities	5052	631	634	22204	2514	3035	23326	5902	5613
#types of entities	1	1	1	7	7	7	4	4	4
# Nested entities	-	-	-	10149	1092	1417	-	-	-
# Discontinuous entities	543	64	68	-	-	-	-	-	-

Table 6: Statistics of the three benchmark NER datasets.

recently NAG is rapidly closing the performance gap against AG models via novel NAG-style pre-training (Qi et al., 2020; Bao et al., 2022). NAG is also applied to other generation tasks like summarizaiton (Li et al., 2022), automatic speech recognition (Deng et al., 2022).

C Datasets

C.1 Open-sourced benchmark datasets

Discontinuous NER datasets We follow Dai et al. (2020); Yan et al. (2021) to use the CADEC dataset[5] in our experiment. Since only the Adverse Drug Events (ADEs) entities have discontinuous annotation, only this type of entity is considered and the other 4 types of entities are discarded.

Nested NER datasets For Nested NER subtask, we adopt the ACE2004[6] dataset. This dataset contains corpuses of newswire, broadcast news and telephone conversations. It contains 7 entity categories: "PER", "ORG", "LOC", "GEP", "VEH", "WEA" and "FAC". In experiment conducted on ACE2004, we use the same data split as Muis and Lu (2017); Yu et al. (2020a), the ratio between train, development and test set is 8:1:1.

Flat NER datasets We adopt the CoNLL03 (Sang and Meulder, 2003) datasets. It is a flat NER dataset with a news corpus and has annotated 4 types of entities as "PER", "LOC", "ORG" and "MISC". For CoNLL03, we follow Lample et al. (2016); Yu et al. (2020a) to train our model on the concatenation of the train and development sets.

C.2 Our proprietary datasets

In this work, we run experiments on two of our proprietary datasets we collect to develop our information extraction or question answering systems.

Chinese medical entity (CME) dataset This dataset is collected from medical records. The collection of these medical records are agreed by the owner, and the data are completely anonymized before being used by the data scentists. This dataset contains 15 entity types, and contains both nested and discontinuous datasets.

Query entity recognition (QER) dataset This dataset is collected from queries from an online question-answering system. The data collection is agreed by all the users. This dataset considers 7 types of entities and it is a flat NER task.

Statistics of the two datasets are listed in Tabel 7.

D Appendix for experimental settings

D.1 Hyper-parameters settings

We run our experiments on NVIDIA Tesla V100 GPUs. The maximum entity length l_{max} is set to 8 for all the three Enghlish benchmark datasets, and 16 for our two proprietary tasks. The maxmimum number of entities starting from the same word O_{max} is set to 5. We manually tune the hyper-parameters including maximum learning rate (max-LR), epochs, maximum tokens per batch, dropout rate, threshold τ_s for each dataset. Specifically, we trial different values of each hyper-parameter within the hyper-parameter search space for ten times and the hyper-parameter values that results in the best performance on the development set are chosen. The search space of each hyper-parameter and the final hyper-parameter configuration are reported in Table 8.

E Appendix: Introduction to discriminative NER models

Models for Discontinuous NER Tang et al. (2018) use LSTM-CRF to recognize continuous and discontinuous adverse drug reaction mentions. Dai et al. (2020) is a transition-based method that utilizes shift-reduce parsers to identify discontinuous entities. Wang et al. (2021b) solve discontin-

[5] https://data.csiro.au/collection/10948v003
[6] https://catalog.ldc.upenn.edu/LDC2005T09

Statistics	*CME*			*QER*		
	Train	Dev	Test	Train	Dev	Test
# Sentences	84328	10540	10540	54390	6800	6800
Avg sent. length	231	232	232	14.3	14.1	14.5
# Entities	1813052	227664	227652	70712	8845	8862
#types of entities	15	15	15	7	7	7
# Nested entities	112458	13463	12968	-	-	-
# Discontinuous entities	63281	6590	6438	-		

Table 7: Statistics of the two proprietary NER datasets.

Hyper-param	Search space	CADEC	ACE2004	CoNLL03	CME	QER
Epochs	{ 30, 50, 75 }	75	50	75	30	50
Max-LR	{1e-5, 2e-5, 5e-5, 1e-4}	2e-5	5e-5	2e-5	1e-5	2e-5
batch size	{1, 2, 4, 8, 16}	4	8	4	16	16
dropout rate	{0.1, 0.2, 0.3, 0.5}	0.3	0.3	0.1	0.1	0.2
τ_s	{0.2, 0.3, 0.5, 0.7, 0.9}	0.2	0.5	0.3	0.5	0.3

Table 8: The hyper-parameter settings for each task in our experiments.

uous NER via the maximal clique discovery algorithm based on graph theory.

Models for Nested NER Yu et al. (2020b) formulate NER as the dependency parsing task. Li et al. (2020) adopt the pointer-based span extraction strategy widely adopted in machine reading comprehension (Zeng et al., 2020). Xu et al. (2021) treat nested NER tasks as multi-class classification of spans and solve it with a multi-head self-attention mechanism. Shen et al. (2021) is a two-stage entity extraction model which first generates candidate spans and then labels the boundary-adjusted span proposals with the corresponding categories.

Models for Flat NER Akbik et al. (2019) dynamically aggregate contextualized embeddings of each encountered string and use a pooling operation to obtain a contextualized word representation from all contextualized instances. Yu et al. (2020b) and Li et al. (2020) can be used to solve both nested and flat NER tasks. Wang et al. (2021a) use the input sentence as a query to retrieve external contextual information with a search engine and concatenate the sentence with external contexts.