Span-based Named Entity Recognition using Attention-based Vector Aggregation

Anonymous ACL submission

Abstract

Named entity recognition (NER) is a fundamental task in natural language processing and is widely implemented in natural language processing systems. Conventionally, the most common method for NER has been sequential labelling utilizing conditional random fields (CRFs). In recent years, span-based methods that input a group of multiple words and determine whether the input group is an NE have also been used. In this paper, we propose a span-based NER method that utilizes attention-based aggregation to combine information from multiple words into a single vector. Experimental results demonstrate that our method achieves a state-of-the-art performance for the WNUT-16 and WNUT-17 datasets commonly used in NER.

1 Introduction

Named entity recognition (NER) is a fundamental task in natural language processing. NER methods extract and classify named entities (NEs) such as person, place, and organization names from input sentences. NER is utilized in many systems that handle natural language, such as for information extractions (Weston et al., 2019; Kumar and Starly, 2022) and personal data anonymisation (Hassan et al., 2019; Sotolář et al., 2021).

The main challenge with NER is that new entities are encountering on a daily basis. To achieve NER that corresponds to new terms, it is necessary to employ classification methods that do not rely solely on the surface of words. In NER utilizing neural networks, it is common to use sequential labelling with bidirectional long short-term memories (LSTMs) and conditional random fields (CRFs) (Panchendrarajan and Amaresan, 2018) to determine whether each word in the input constitutes an NE or part of an NE (Figure 1-(a)). In recent years, span-based NER methods have also been used (Luan et al., 2018; Lin et al., 2019). Figure 1-(b) outlines the span-based approach. A single vector is created from a sequence of words targeted for classification and is used to determine whether the sequence constitutes an NE and, if so, the type of NE. Since there are various length of tokens in the target span, it is necessary to convert the vectors of different lengths into a single vector. In conventional methods, LSTM and max pooling are often used. However, LSTM is prone to being strongly influenced by the vectors of tokens input at the beginning or end, and also, with max pooling, there is an issue where the output vectors can become similar for sequences with many input words (e.g., between "The Times" and "The Times of London").

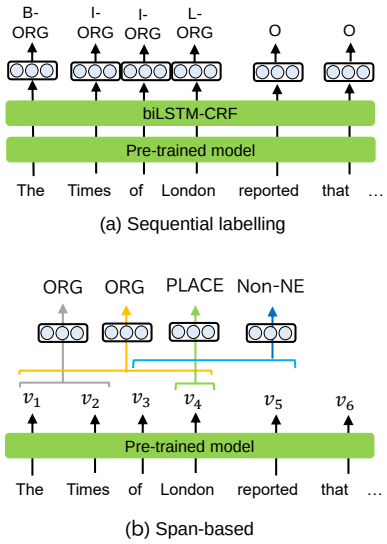

Figure 1: Two types of NER methods. In sequential labeling, labels such as beginning (B-), intermediate (I-), last (L-), or non-NE (O) as well as NE types are assigned to each word. In span-based methods, vectors representing spans of words are calculated, and it is determined whether each entire span constitutes an NE.

To address these issues, we propose a method that utilizes multi-head attention in the vector aggregation part instead of LSTM or max pooling.

In our proposed approach, we input the entire sentence into a pre-trained model to obtain a sequence of vectors. From these vectors, we use multi-head attention to add up the vectors corresponding to words in the target span. This enables the aggregation of inputs of varying lengths into a single vector that can be utilized for the NER task. We then feed these aggregated vectors into feed-forward neural networks (FFNNs) to perform NER. The proposed method has achieved a performance surpassing that of the conventional state-of-the-art (SoTA) on several commonly used NER datasets.

Our contributions are summarized as follows. (1) We propose a simple span-based NER method utilizing attention-based vector aggregation. (2) The proposed method shows a good performance for informal text such as found on social network sites (SNSs), and achieves a SoTA performance on multiple datasets targeting SNSs. (3) The proposed method achieves a good performance in classifying unknown NEs that do not appear in the training data, which is a useful feature for NER because training data cannot cover all NEs. (4) The proposed method performs well regardless of the number of words constituting an NE.

2 Related Work

2.1 NER Tasks

NER tasks can be classified into three subtasks: flat, nested, and discontinuous NER. Flat NER, which is our target, is a traditional task with a long history of study (Borkowski and Watson, 1967), while nested (Finkel and Manning, 2009) and discontinuous NER (Tang et al., 2013) are newer tasks.

In flat NER, nested structures of NEs are not considered, so every word is considered to belong up to one NE. In contrast, nested and discontinuous NER do consider nested structures, so every word can belong to multiple NEs. Nested NER only considers NEs that consist of continuous words, whereas discontinuous NER considers NEs that consist of both continuous and distanced words. The differences between these tasks are summarized in Figure 2.

We believe that for real-world applications such as anonymisation, the flat NER model, which only assigns one label to each word and does not require manual selection, is better than the nested NER model, which outputs multiple labels for each word. Therefore, this paper focuses on flat NER.

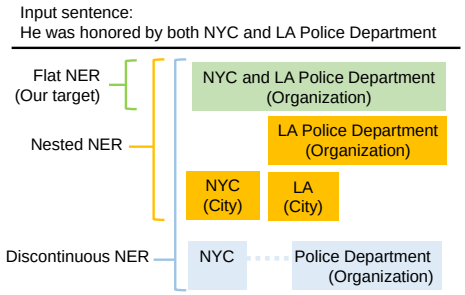

Figure 2: Comparison of three NER tasks. Flat NER considers only the NE in the green box. Nested NER considers the nested structures of NEs, so NEs in the orange boxes are also considered. Discontinuous NER considers NEs consisting of distanced words, so the NE in the blue box is also considered.

Sequential Labelling-based Methods

Many methods for flat NER are based on sequential labelling using CRFs (Huang et al., 2015; Lample et al., 2016). These methods apply bidirectional LSTMs (bi-LSTMs) before CRFs to obtain word vectors that take the context of the sentence into account, and then classify using the word vectors whether the corresponding word belongs to NE.

Generally, the vocabulary of NE is large, and many new NEs are appearing all the time. Therefore, several NER methods to improve the handling of unknown NEs in training data have been proposed. Akbik et al. (2019) proposed utilizing contextual string embeddings in a character language model and then feeding the character-based embedded vectors into bi-LSTM to obtain word-level vectors, thereby improving the classification quality of low-frequency words in training data.

Several NER methods utilizing external context have been proposed. Yamada et al. (2020) proposed language understanding with knowledge-based embeddings (LUKE), which is transformer-based pre-trained representations of words and entities taking document-level contexts into account. Wang et al. (2021b) proposed a method that uses retrieved related contexts from a search engine as external contexts of the inputs. They also proposed cooperative learning to improve the robustness of the models when no external contexts are available.

Although external data is useful for NER, methods that utilize external data tend to complicate the model and lengthen the calculation time. To consider real-world usage, we opt for a method that does not utilize external data.

Span-based Methods

The methods above utilize sequential labelling-based methods, and while they have obtained good results, they can be utilized only for flat NER. Recently, many methods have been developed that can be used for nested NER. Luan et al. (2018) proposed a multi-task model that identifies NEs, relations, and co-reference clusters in scientific articles by using span representing vectors. Lin et al. (2019) proposed anchor-region networks, which comprise an anchor detector for extracting anchor words that can be the head word of NEs and a region detector for detecting the span of the NEs that includes the anchor word. Wan et al. (2022) proposed a method that builds an entity-entity graph and a span-entity graph based on n-gram feature similarity from training data. They feed the graph into graph convolutional networks to obtain better span representations.

The above methods are called span-based methods, as they determine whether each span of one or more words is an NE.

Other Methods

There are other NER methods in addition to the sequential labelling-based and span-based ones. Ju et al. (2018) proposed a method that utilizes a layered sequential labeling-based network to capture nested NEs. Katiyar and Cardie (2018) proposed a hypergraph-based NER method that uses scores calculated by LSTM as edge probabilities.

Recently, new approaches for NER that take advantage of improved computational performance and advances in neural network, have been proposed such as diffusion model-based (Shen et al., 2023a; Ma et al., 2023), prompt learning-based (Shen et al., 2023b; Chen et al., 2023), and contrastive and prototyping learning-based (Das et al., 2022; Zhou et al., 2023) NER.

3 Our NER Method

3.1 General Span-based NER

In span-based NER methods, NEs are extracted by simultaneously determining the span that represents the range of the NE and its type. Some methods utilize separate networks for span estimation and entity type classification (Lin et al., 2019; Tan et al., 2020), while other input all possible patterns of consecutive words from the input sentence as spans to a model, and then make determinations on whether the input span is an NE and, if so, classify its type using a single network (Wan et al., 2022). In this paper, we employ the latter.

The first step involves vectorizing each word in the input text using pre-trained models. Subsequently, all possible spans consisting of consecutive multiple words within the input text are created. The vectors of the words included in each span are aggregated to form a vector representing the entire span. This vector is then input to FFNNs to determine whether the span constitutes an NE and, if so, to classify its type. Max/avg pooling and LSTMs are commonly utilized for vector aggregation.

3.2 Proposed Method

3.2.1 Model structure

The proposed method utilizes a multi-head attention for aggregating vectors. Its model structure is shown in Figure 3, in which the input sentence consists of $\{w_0, w_1, \cdots w_{|w|}\}$, where $|w|$ represents the number of words in the input sentence, and the target span to be classified is w_k and w_{k+1}.

The input sentence is first tokenized into $t_{0,1}, \cdots t_{k,1}, \cdots, t_{k,|t_k|}, \cdots$, where $t_{i,j}$ represents the j-th token of w_i and $|t_k|$ represents the number of token in word w_k. The tokens are vectorized utilizing a pre-trained model. Then, the token-representing vectors of the target span are fed into a transformer-decoder-like block (denoted as *decoder*). The multi-head attention-based vector aggregation is done in this block, and the aggregated vector is then fed to the FFNN of the *decoder*. We add the vector from residual connections and apply layer normalization in the "add & norm" block as done in transformer. After calculating the N-layer of the *decoder*, the output vector is fed to 3-layer FFNNs to obtain the NER label for the span.

3.2.2 Attention-based Aggregation

Our method extracts vectors output from the target span and feeds them into the multi-head attention of the *decoder* as key and value. It also prepares a query vector of the multi-head attention of the *decoder*. We utilize a ⟨CLS⟩ representing vector that embeds single ID ⟨CLS⟩ using a learnable embedding layer as the query vector. ⟨CLS⟩ is a special token, which is the same as the ⟨CLS⟩ in BERT (Devlin et al., 2019). The ⟨CLS⟩ representing vector enables multi-head attention to aggregate vectors into a single vector that is suitable for a subsequent FFNN layer. In the learning steps, we expect that the ⟨CLS⟩ representing vector is

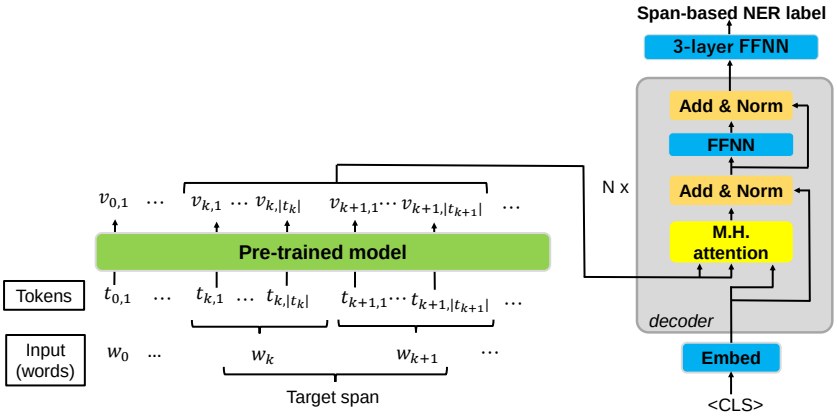

Figure 3: Model structure of proposed method.

learned to be a better vector for NER in the output of multi-head attention.

We borrowed the idea of ⟨CLS⟩ from BERT, but unlike BERT, we used a transformer decoder-like block. This is because we want to reduce the effect of the ⟨CLS⟩ representing vector from the output of the multi-head attention. The attention mechanism is calculated as:

$$\text{Attention} = \text{Softmax}(\frac{QK^T}{\sqrt{D}})V, \qquad (1)$$

where Q, K, and V are query, key, and value vector, and D represents the dimension of the input vector. Note that, for clarity, we eliminated the head indicator and the projection matrices applied to Q, K, and V. In the equation, we first calculate the attention weight using Q and K, and then calculate the weighted sum of V while using the attention weight. If the inputs of Q, K, and V are all the same vector as self-attention, the attention weight will be high for the same token, of course including ⟨CLS⟩. Therefore, the final output for the token ⟨CLS⟩ is highly affected by the ⟨CLS⟩ representing vector. On the other hand, our method does not include ⟨CLS⟩ representing vector in K and V, so the final output is less affected by it. Obviously, the ⟨CLS⟩ representing vector itself is not useful for NER because we use all the same vectors regardless of the target span. We believe that reducing the effect from the ⟨CLS⟩ representing vector is effective in this model.

The ⟨CLS⟩ representing vector itself may not directly contribute to NE classification, but it is added with output of attention in the *decoder* through a residual connection. While removing the residual connection passing through the first layer of attention could reduce the influence of the ⟨CLS⟩ vector, the results of our preliminary experiments showed that using the residual connection in the first layer had a better performance. This is presumably because, during training, the presence of the residual connection enabled the embedding layer to better train the embedding of the ⟨CLS⟩ vector.

3.2.3 Training

In the model training, the first step involves tokenizing the input sentence and vectorizing it with a pre-trained model. Next, spans corresponding to NE are extracted from the gold labels, and non-NE spans are created as negative examples. Negative examples are limited to a maximum of ten words. Since training on all negative examples can be computationally expensive, 10% of all negative examples are randomly sampled. For these spans, our *decoder* block is applied, and the output of the *decoder* is fed into the 3-layer FFNNs. Then, the model calculates a loss, and the model is updated using the loss.

During training, a single loss is utilized, and separate optimizers with different learning rates are employed for fine-tuning the pre-trained model and training the *decoder* and 3-layer FFNNs.

3.2.4 Decoding

In flat-NER, each word in the input is treated as belonging to up to one NE. In contrast, the proposed method allows each word to belong to multiple NEs. Therefore, our model outputs the NE with the maximum score and evaluates the method as a flat-NER approach.

Our method outputs the score for the type of NE or non-NE for each span created from the input text using the trained model. If the score for non-NE is the highest, we discard this span. Otherwise, we output the score along with the type of NE with

	Train	Dev	Test	Labels
WNUT-16	2,394	1,000	3,849	10
WNUT-17	3,394	1,009	1,287	6
CoNLL 2003	14,987	3,466	3,684	4
CoNLL ++	14,987	3,466	3,466	4

Table 1: Statistics of datasets.

the maximum score. We arrange these outputs in descending order based on scores.

Starting from the span with the highest score, we assign labels to the words in the span with NE labels. If the span contains any words already labeled by a span with a higher score, we discard that span. When assignment is completed for all spans, words without labels are considered non-NE, and the final result of NER is then output.

This decoding method is generally referred to as a greedy algorithm. While other methods (such as beam search, which is often considered to have a better performance compared to the greedy algorithm) are commonly used, we adopted the greedy algorithm in this study because preliminary experiments showed that it performed better. A comparison between the greedy algorithm and beam search is provided in Section 4.5.5.

4 Experiments

4.1 Data

We utilized four NER datasets: WNUT-16 (Strauss et al., 2016), WNUT-17 (Derczynski et al., 2017), CoNLL 2003 (Tjong Kim Sang and De Meulder, 2003) and CoNLL ++ (Wang et al., 2019). WNUT-16 and WNUT-17 are collected from social media posts, so they include colloquial languages, abbreviations, and many unique NEs. As a contrast, CoNLL 2003 consists of news articles, so it includes polite and well-written sentences. CoNLL ++ is a modified version of CoNLL 2003 that includes a collection of mislabels from the test data of CoNLL 2003. The statistics of the datasets are listed in Table 1.

4.2 Experimental Settings

We utilized PyTorch (Paszke et al., 2019) and Transformers (Wolf et al., 2020) for models implementation, and we optimized the models using the RAdam optimizer (Liu et al., 2020). We utilized XLM-RoBERTa-large (Conneau et al., 2019) as a pre-trained model. We set the number of layers of our *decoder* to two and the number of heads in the multi-head attention to eight for all datasets. Learning rates were tuned for each dataset, and set to 2.0×10^{-4} for WNUT-16 and WNUT-17 and to 1.0×10^{-4} for the others. The learning rate for fine-tuning of the pre-trained model was set to 1.0×10^{-5} for WNUT-16, 2.0×10^{-5} for WNUT-17, and 5.0×10^{-6} for the others. We utilized a linear scheduler and warmed up with three epochs. The models were trained with the batch size of 50 and the number of training epochs of 50. We utilized focal loss (Lin et al., 2017) with 2.0 of γ for model training. We evaluated the model in each epoch using development data, and then chose the model with the best micro-F1 score. In total, we trained each model five times with different random seeds. We set the dropout ratio to 0.1 for fine-tuning of pre-trained model, 0.3 for the components in the *decoder*, and 0.5 for the 3-layer FFNN. For evaluation, we used seqeval (Nakayama, 2018) to calculate the F1 scores.

4.3 Baseline Methods

We prepared four in-house baseline methods and four conventional methods that achieved SoTA results in each dataset.

4.3.1 In-house Baseline Methods

To investigate the effectiveness of our attention-based aggregation of vectors, we prepared four in-house baseline methods. As shown in Figure 4, they utilize transformer, max/average pooling, or bi-LSTM instead of attention-based aggregation. Transformer, pooling, and bi-LSTM in each method accept vectors of tokens of the target span from the pre-trained model, aggregate them into a single vector, and feed them to 3-layer FFNN, which is the same structure as with proposed method. The only difference is how the vectors are aggregated.

In the transformer baseline, we utilized ⟨CLS⟩ vectors. The ⟨CLS⟩ embedding vector is concatenated with vectors from the pre-trained model, and the concatenated vector is then fed into the transformer encoder model. The baseline method extracts a single vector that corresponds to ⟨CLS⟩ vector, and the corresponding vector is fed into the FFNN layer to classify whether it is an NE.

Note that, in order to match the number of dimensions of the FFNN with that of other models, bi-LSTM calculates by halving the vector size from the output of the pre-trained model in the hidden layer and a vector that concatenated the hidden states of both directions are fed to the FFNN.

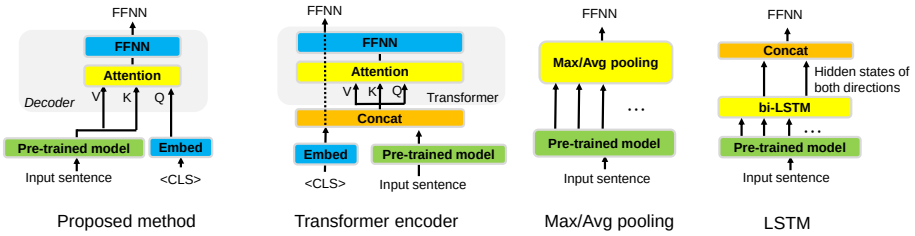

Figure 4: In-house baseline methods and our proposed method. All methods accept vectors of tokens of the target span from the pre-trained model and then aggregate them into a single vector. For clarity, we eliminated residual paths and "add & Norm" blocks from both transformer and proposed in this figure.

	WNUT16		WNUT17		CoNLL 2003		CoNLL ++	
	Median	Average	Median	Average	Median	Average	Median	Average
Proposed and in-house baselines methods								
Proposed	**60.68**	**60.21** ±0.97	**61.64**	**61.29** ±0.70	93.06	93.03 ±0.11	94.42	94.39 ±0.19
Max pooling	59.38	59.43 ±0.53	59.88	59.95 ±0.78	93.21	93.21 ±0.08	94.35	94.27 ±0.22
Avg. pooling	59.85	59.56 ±0.98	57.88	57.86 ±0.72	93.01	93.05 ±0.16	94.27	94.26 ±0.10
LSTM	60.03	60.01 ±0.20	61.51	60.72 ±1.31	93.14	93.14 ±0.08	94.58	94.55 ±0.16
Transformer	60.23	**60.21** ±0.51	61.27	61.20 ±0.90	93.23	93.23 ±0.11	94.51	94.41 ±0.29
Current SoTA methods								
HGN (XLNET)	59.50*	–	57.41	–	–	–	–	
CL-KL	58.98	–	60.45*	–	93.56	–	94.81	–
ACE+document-context	–	–	–	–	**94.60***	–	–	–
Noise-robust Co-regularization	–	–	–	–	94.22	–	**95.88***	

Table 2: Experimental results. * means current (as is in January, 2024) state-of-the-art results. For our methods, we show median, average and standard deviation of the five attempts with different random seeds. Underlined means outperformed SoTA. **bold** means the best result in the table.

We use the same parameters as the proposed method in the experiments. For the transformer baseline, we utilized 2-layer transformer for WNUT-16 and CoNLL ++, and 3-layer for the others. For the LSTM baseline, we used 2-layer bi-LSTM for WNUT-17 and 1-layer bi-LSTM for the others. These parameters are set on the basis of the results of the development data. Dropout ratio for the transformer encoder and bi-LSTM are set to 0.3.

4.3.2 Current SoTA models

HGN (Hu et al., 2022) This is the current SoTA on the WNUT-16 dataset. The method utilizes pre-trained model-based embedding (hero module) and character-based recurrent neural networks (gang module), which are used to capture global and local features, respectively. The features are gathered in a multi-window attention module, which also considers the relative position information in the module.

CL-KL (Wang et al., 2021c) This is the current SoTA on the WNUT-17 dataset. The method utilizes external contexts retrieved using a search engine, and the external contexts are then fed into the model with the target context. The method uses two losses during the training process, one with external context and one without, and combines them using a KL divergence-based method to achieve cooperative learning.

ACE (Wang et al., 2021a) This is the current SoTA on the CoNLL-03 dataset. The method automatically searches for a better embedding concatenation using reinforcement learning strategies.

Noise-robust Co-regularization (Zhou and Chen, 2021) This is the current SoTA on the CoNLL ++ dataset. The method consists of several neural models with identical structures but different parameter initializations to prevent overfitting on noisy labels.

4.4 Experimental Results

Experimental results are given in Table 2. As shown, our proposed method achieved the SoTA result in the WNUT-16 and WNUT-17 datasets.

	WNUT-16		CoNLL 2003		CoNLL ++	
	Unknown	Known	Unknown	Known	Unknown	Known
No. in test data	1,675	363	1,157	2,417	1,171	2,399
Proposed	61.13	83.47	**90.41**	95.99	**91.89**	97.17
Max pooling	**61.91**	82.09	90.49	**95.70**	91.20	97.04
Avg pooling	59.88	81.27	**90.41**	95.66	91.20	97.12
LSTM	56.60	**83.74**	89.46	**95.70**	91.03	97.17
Transformer	60.54	82.92	**90.41**	95.66	91.37	**97.33**

Table 3: Comparison of correct rate between known and unknown NEs. Note that the results in this table do not take recall or precision into account. This result is based on the model with median performance in Table 2.

	No. of words in NE				
	1 word	2 words	3 words	4 words	5 or more words
No. in test data	718	220	74	37	30
Proposed	61.56	62.72	**36.49**	16.22	**13.33**
Max pooling	**64.07**	61.82	28.38	16.22	10.00
Avg pooling	52.79	60.91	29.73	16.22	10.00
LSTM	56.96	**65.45**	**36.49**	16.22	6.67
Transformer	62.67	63.64	32.43	16.22	**13.33**

Table 4: Correct rate based on length of NEs in WNUT-17 dataset. Note that the results in this table do not take recall or precision into account. This result is based on the model with median performance in Table 2. Results for other datasets are provided in the Appendix.

Also, many of our in-house baseline methods outperformed current SoTA in the WNUT-16 and WNUT-17 datasets. However, in the CoNLL 2003 and CoNLL ++ datasets, our proposed and in-house baseline methods could not outperform the current SoTA.

4.5 Discussion

4.5.1 Superiority of Span-based NER

All of our proposed and in-house baseline methods are span-based NER methods. The fact that many of these methods surpassed the current SoTA in WNUT-16 and WNUT-17 indicates that span-based methods may be more useful for NER compared to sequential labelling methods especially for informal language such as social media posts, which tend to contain many omissions and misspellings. Sequential labelling methods often degrade when such issues arise, but in span-based methods, the impact of such issues is presumably mitigated during the process of creating the vector for the span. In addition, among span-based methods, the proposed method using attention-based aggregation had the best performance on the WNUT-16 and WNUT-17 datasets, which confirms that our attention-based aggregation is particularly effective for such informal language.

On the other hand, in well-structured texts such as those in CoNLL 2003 and CoNLL ++, performance degradation in sequential labelling is less likely to occur. This presumably why the traditional methods using sequential labelling outperformed the span-based methods for these texts.

4.5.2 Performance for Unknown NEs

WNUT-17 is designed so that all NEs appearing in the test data are unknown and do not appear in the training data. The high performance achieved on this dataset indicates that the proposed method performs well with unknown NEs. We compared the performance by separating known and unknown NEs in other datasets using the proposed and in-house baseline methods. The results are presented in Table 3. While the proposed method had a good performance for unknown NEs, other methods often outperformed it for known NEs. This is presumably why the proposed method sometimes could not outperform the in-house baselines in datasets with many known NEs, such as CoNLL 2003.

Looking the performance with transformer, we can see that the proposed method has a better performance for unknown NEs. This is because when an unknown NEs is input, the attention weight for the ⟨CLS⟩ corresponding vector of the self-attention in the transformer tends to be higher for the ⟨CLS⟩ itself, and as a result, the influence of the ⟨CLS⟩ embedding vector on the output of the transformer becomes large. This problem does not occur with the proposed method because ⟨CLS⟩ is not included in the attention key and value, which is presumably why the proposed method achieved a high performance for unknown NEs.

	⟨CLS⟩	Max pooling	Avg. pooling
WNUT-16	**60.68**	59.79	60.15
WNUT-17	**61.64**	61.24	61.23
CoNLL 2003	93.06	**93.16**	92.81
CoNLL ++	94.42	94.37	**94.54**

Table 5: Comparison between ⟨CLS⟩ and pooling.

Data	Greedy	Beam search
WNUT-16	**60.68**	60.39
WNUT-17	**61.64**	61.48
CoNLL 2003	**93.06**	92.97
CoNLL ++	94.42	**94.53**

Table 6: Comparison between greedy algorithm and beam search.

4.5.3 Differences in Performance due to NE Length

We also examined the effect of different lengths of NEs. The results are provided in Table 4.

LSTM is notably weak for single-word NEs, while max pooling performs well with single-word NEs but poorly with longer ones. This is because, LSTM's initial hidden state, which is constant regardless of input and is not useful for NER, has a significant impact when the input consists of one token, making it more prone to classification errors. On the other hand, max pooling simply takes the maximum value of vectors, resulting in a lower performance when the number of input tokens is high. Both the proposed method and transformer utilize an attention mechanism for vector aggregation, allowing for complex computations than simple max pooling. Additionally, since the proposed method does not have an initial hidden state like LSTM, it works well even with only one token input. Overall, the proposed method and transformer achieved a good performance on average regardless of the number of words composing the NE.

4.5.4 Effectiveness of ⟨CLS⟩

One of the key characteristics of the proposed approach is its use of ⟨CLS⟩ as input to the decoder. To examine the effect of ⟨CLS⟩, we conducted experiments using max pooling and average pooling instead of a ⟨CLS⟩ representing vector. The results are presented in Table 5, showing the median of five experiments for each method. As shown, using ⟨CLS⟩ performed the best for the WNUT-16 and WNUT-17 datasets, and was close to the best on the other datasets. When max/avg pooling is used as the input to the decoder instead of a ⟨CLS⟩ representing vector, it is possible to output a vector close to the whole input span, but it may not necessarily be a useful vector for NER. On the other hand, in the proposed approach, ⟨CLS⟩ is flexibly learned, allowing for the extraction of a vector useful for NEs detection regardless of the input span, which is presumably what contributed to the better performance compared to using max/avg pooling.

4.5.5 Greedy Algorithm vs Beam Search

We employed a greedy algorithm for decoding. Typically, more competitive methods such as beam search are used in such decoding processes, but, in our experiments, the performance was actually better with using greedy algorithm. Table 6 presents the results of greedy algorithm and beam search with proposed method. The models were trained simultaneously, and we evaluated the model for each epoch using both the greedy algorithm and beam search on the development set. The models that achieved the best performance for each decoding method were then utilized for the evaluation on the test set. We show the median of five experiments.

We attribute the underperformance of beam search in this study to the difficulty in setting scores for non-NE cases. Since a single word can be included in up to ten spans, it is difficult to decide from which span the non-NE score should be used. In beam search, we utilized the minimum score for a word among all non-NE scores of spans containing the word, which is the best setting in our experiments. However, this approach tended to favor longer spans as NEs, resulting in an overall degradation in performance. The greedy algorithm was able to outperform beam search thanks to avoiding the use of non-NE scores.

5 Conclusion

We proposed a simple span-based NER method that utilizes attention-based vector aggregation in a *decoder* block. Our method performed particularly well for informal text, and achieved SoTA results on the SNS-based datasets of WNUT-16 and WNUT-17. Moreover, it demonstrated a good performance in extracting unknown NEs that did not appear in the training data and it performed well regardless of the number of words in the NEs.

In future work, we will investigate how to achieve robust classification for NEs appearing in the training data and explore better search methods than the greedy algorithm.

Limitations

In this study, we reported results using only XLM-RoBERTa as a pre-trained model. While the overall performance can be influenced by the pre-trained model, we believe we can achieve a good performance with other models. However, we have not conducted experiments with more recent and complex models due to constraints in terms of computational resources and time.

In this paper, we conducted experiments using only the English dataset, but we believe that it can be applied to other languages as it does not use language-dependent parameters.

Our method is designed to be applicable to nested NER. However, we did not conduct any experiments using nested NER datasets in this study, again due to constraints in terms of computational resources and time.

References

Alan Akbik, Tanja Bergmann, and Roland Vollgraf. 2019. Pooled contextualized embeddings for named entity recognition. 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 724–728, Minneapolis, Minnesota. Association for Computational Linguistics.

Casimir Borkowski and Thomas J. Watson. 1967. An experimental system for automatic recognition of personal titles and personal names in newspaper texts. In *COLING 1967 Volume 1: Conference Internationale Sur Le Traitement Automatique Des Langues*.

Yanru Chen, Yanan Zheng, and Zhilin Yang. 2023. Prompt-based metric learning for few-shot NER. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7199–7212, Toronto, Canada. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *CoRR*, abs/1911.02116.

Sarkar Snigdha Sarathi Das, Arzoo Katiyar, Rebecca Passonneau, and Rui Zhang. 2022. CONTaiNER: Few-shot named entity recognition via contrastive learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6338–6353, Dublin, Ireland. Association for Computational Linguistics.

Leon Derczynski, Eric Nichols, Marieke van Erp, and Nut Limsopatham. 2017. Results of the WNUT2017 shared task on novel and emerging entity recognition. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 140–147, Copenhagen, Denmark. 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: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jenny Rose Finkel and Christopher D. Manning. 2009. Nested named entity recognition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 141–150, Singapore. Association for Computational Linguistics.

Fadi Hassan, David Sánchez, Jordi Soria-Comas, and Josep Domingo-Ferrer. 2019. Automatic anonymization of textual documents: detecting sensitive information via word embeddings. In *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 358–365. IEEE.

Jinpeng Hu, Yaling Shen, Yang Liu, Xiang Wan, and Tsung-Hui Chang. 2022. Hero-gang neural model for named entity recognition. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1924–1936, Seattle, United States. Association for Computational Linguistics.

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

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: Human Language Technologies, Volume 1 (Long Papers)*, 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: Human Language Technologies, Volume 1 (Long Papers)*, pages 861–871, New Orleans, Louisiana. Association for Computational Linguistics.

Aman Kumar and Binil Starly. 2022. fabner: information extraction from manufacturing process science domain literature using named entity recog-

nition. *Journal of Intelligent Manufacturing*, 33(8):2393–2407.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. 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.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2020. On the variance of the adaptive learning rate and beyond. In *Proceedings of the Eighth International Conference on Learning Representations (ICLR 2020)*.

Shuheng Liu and Alan Ritter. 2023. Do CoNLL-2003 named entity taggers still work well in 2023? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8254–8271, Toronto, Canada. Association for Computational Linguistics.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3219–3232, Brussels, Belgium. Association for Computational Linguistics.

Tingting Ma, Qianhui Wu, Huiqiang Jiang, Börje Karlsson, Tiejun Zhao, and Chin-Yew Lin. 2023. CoLaDa: A collaborative label denoising framework for cross-lingual named entity recognition. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5995–6009, Toronto, Canada. Association for Computational Linguistics.

Hiroki Nakayama. 2018. seqeval: A python framework for sequence labeling evaluation. Software available from https://github.com/chakki-works/seqeval.

Rrubaa Panchendrarajan and Aravindh Amaresan. 2018. Bidirectional LSTM-CRF for named entity recognition. In *Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation*, Hong Kong. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023a. DiffusionNER: Boundary diffusion for named entity recognition. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3875–3890, Toronto, Canada. Association for Computational Linguistics.

Yongliang Shen, Zeqi Tan, Shuhui Wu, Wenqi Zhang, Rongsheng Zhang, Yadong Xi, Weiming Lu, and Yueting Zhuang. 2023b. PromptNER: Prompt locating and typing for named entity recognition. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12492–12507, Toronto, Canada. Association for Computational Linguistics.

Ondřej Sotolář, Jaromír Plhák, and David Šmahel. 2021. Towards personal data anonymization for social messaging. In *International Conference on Text, Speech, and Dialogue*, pages 281–292. Springer.

Benjamin Strauss, Bethany Toma, Alan Ritter, Marie-Catherine de Marneffe, and Wei Xu. 2016. Results of the WNUT16 named entity recognition shared task. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, pages 138–144, Osaka, Japan. The COLING 2016 Organizing Committee.

Chuanqi Tan, Wei Qiu, Mosha Chen, Rui Wang, and Fei Huang. 2020. Boundary enhanced neural span classification for nested named entity recognition. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 9016–9023.

Buzhou Tang, Hongxin Cao, Yonghui Wu, Min Jiang, and Hua Xu. 2013. Recognizing clinical entities in hospital discharge summaries using structural support vector machines with word representation features. In *BMC medical informatics and decision making*, volume 13, pages 1–10. BioMed Central.

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.

Juncheng Wan, Dongyu Ru, Weinan Zhang, and Yong Yu. 2022. Nested named entity recognition with span-level graphs. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*

(Volume 1: Long Papers), pages 892–903, Dublin, Ireland. Association for Computational Linguistics.

Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. 2021a. Automated concatenation of embeddings for structured prediction. 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 2643–2660, Online. Association for Computational Linguistics.

Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. 2021b. Improving Named Entity Recognition by External Context Retrieving and Cooperative Learning. In *the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (**ACL-IJCNLP 2021**)*. Association for Computational Linguistics.

Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. 2021c. 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 (Volume 1: Long Papers)*, pages 1800–1812, Online. Association for Computational Linguistics.

Zihan Wang, Jingbo Shang, Liyuan Liu, Lihao Lu, Jiacheng Liu, and Jiawei Han. 2019. CrossWeigh: Training named entity tagger from imperfect annotations. 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 5154–5163, Hong Kong, China. Association for Computational Linguistics.

Leigh Weston, Vahe Tshitoyan, John Dagdelen, Olga Kononova, Amalie Trewartha, Kristin A Persson, Gerbrand Ceder, and Anubhav Jain. 2019. Named entity recognition and normalization applied to large-scale information extraction from the materials science literature. *Journal of chemical information and modeling*, 59(9):3692–3702.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. Luke: Deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6442–6454.

Ran Zhou, Xin Li, Lidong Bing, Erik Cambria, and Chunyan Miao. 2023. Improving self-training for cross-lingual named entity recognition with contrastive and prototype learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4018–4031, Toronto, Canada. Association for Computational Linguistics.

Wenxuan Zhou and Muhao Chen. 2021. Learning from noisy labels for entity-centric information extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5381–5392, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

A Performance Differences due to NE length in datasets other than WNUT-17

Performance differences due to NE length in datasets other than WNUT-17 are shown in Table 7. We can see a similar tendency to WNUT-17 in WNUT-16 and CoNLL ++, but not in CoNLL 2003. CoNLL 2003 is widely utilized for evaluating NER models, but it has been reported that the test set contains some annotation errors (Liu and Ritter, 2023). This is presumably why only CoNLL 2003 showed a different trend. In fact, CoNLL 2003 and CoNLL ++ have the same training data, but CoNLL++ has the same tendency as WNUT-16 and WNUT-17.

B Parameter size

The parameter sizes of the proposed and in-house baselines are shown in Table 8. The parameter size of proposed method is the same as that of transformer baseline, and it is smaller than LSTM baseline.

C Computing Resources and Computing Time

We utilized NVIDIA A100 (80/40 GB) for our experiments. The proposed method took approximately 80 minutes for 50 epochs of training and evaluation with the WNUT-17 dataset, 120 minutes for WNUT-16, and four hours for CoNLL 2003 and CoNLL ++. This is almost the same computing time as the transformer baseline, indicating

	No. of words in NE				
	1	2	3	4	5 or more
	WNUT-16				
No. in test data	2,038	872	344	115	104
Proposed	65.11	65.14	45.93	33.04	18.27
Max pooling	**65.51**	63.88	45.06	26.96	11.54
Avg pooling	63.69	**68.12**	47.96	**37.39**	14.42
LSTM	61.43	66.06	46.51	32.17	**19.23**
Transformer	64.52	63.88	**52.62**	32.17	13.46
	CoNLL 2003				
No. in test data	3,574	1,776	236	37	25
Proposed	**94.18**	**95.10**	88.13	91.89	92.00
Max pooling	94.01	94.82	90.25	91.89	**100.00**
Avg pooling	93.96	94.76	89.83	91.89	96.00
LSTM	93.68	94.82	**91.11**	91.89	96.00
Transformer	93.96	94.71	90.25	**94.59**	**100.00**
	CoNLL ++				
No. in test data	3,570	1,815	246	43	28
Proposed	**95.43**	94.38	**91.06**	90.70	**89.29**
Max pooling	95.12	**95.32**	90.24	**93.02**	85.71
Avg pooling	95.18	94.77	90.65	**93.02**	85.71
LSTM	95.15	94.99	89.84	90.70	85.71
Transformer	95.38	94.82	89.43	**93.02**	82.14

Table 7: Recalls based on length of NEs in WNUT-16, CoNLL 2003, and CoNLL ++ datasets.

Proposed (2layer)	578M
Max/Avg pooling	562M
LSTM (2layer)	610M
Transformer (2 layer)	578M

Table 8: Parameter size of each method. Note that, this parameter size includes 559M parameters of the pre-trained model.

that the computational cost of our method is almost the same as that of transformer.

D Hyperparameter search

We searched for the best settings for the learning rate and number of layer for *decoder* of the proposed method, transformer, and LSTM baselines. The learning rate for our model was searched from $\{1.0, 2.0, 5.0\} \times 10^{\{-3,-4\}}$ and for fine-tuning, from $\{1.0, 2.0, 5.0\} \times 10^{-4,-5\}}$. The number of layers is searched from 1 to 3.