scientific reports

Check for updates

OPEN

Attention-based interactive multi-level feature fusion for named entity recognition

Yiwu Xu[1] & Yun Chen[2]✉

Named Entity Recognition (NER) is an essential component of numerous Natural Language Processing (NLP) systems, with the aim of identifying and classifying entities that have specific meanings in raw text, such as person (PER), location (LOC), and organization (ORG). Recently, Deep Neural Networks (DNNs) have been extensively applied to NER tasks owing to the rapid development of deep learning technology. However, despite their advancements, these models fail to take full advantage of the multi-level features (e.g., lexical phrases, keywords, capitalization, suffixes, etc.) of entities and the dependencies between different features. To address this issue, we propose a novel attention-based interactive multi-level feature fusion (AIMFF) framework, which aims to improve NER by obtaining multi-level features from different perspectives and interactively capturing the dependencies between different features. Our model is composed of four parts: the input, feature extraction, feature fusion, and sequence-labeling layers. First, we generate the original word- and character-level embeddings in the input layer. Then, we incorporate four parallel components to capture global word-level, local word-level, global character-level, and local character-level features in the feature extraction layer to enrich word embeddings with comprehensive multi-level semantic features. Next, we adopt cross-attention in the feature fusion layer to fuse features by exploiting the interaction between word- and character-level features. Finally, the fused features are fed into the sequence labeling layer to predict the word labels. We conducted generous comparative experiments on three datasets, and the experimental results showed that our model achieved better performance than several state-of-the-art models.

Keywords Named entity recognition, Multi-level features, Cross-attention, Feature fusion

Named Entity Recognition (NER) aims to identify entities with specific meanings from raw text and assign corresponding labels to them, such as person (PER), location (LOC), and organization (ORG)[1]. As an essential component of natural language processing (NLP) systems, NER has been extensively applied to many downstream tasks, such as information retrieval[2], document summarization[3], question-answering systems[4] and language translation[5].

Dictionary-based methods have traditionally achieved entity extraction by matching words in the input text with terms in a dictionary[6]. Although this method is simplistic, the ongoing proliferation of entities and the increasing diversity of symbols have made entity recognition extremely challenging. Rule-based methods, which rely on features designed for each specific task, demonstrate good performance within their domain[7]. Machine learning-based methods can effectively extract entities using various statistical models and algorithms[8]. However, both machine learning-based and rule-based methods have significant drawbacks due to their high dependence on feature engineering. This process is not only time-consuming and labor-intensive but also requires extensive domain knowledge[9].

Recently, Deep Neural Networks (DNNs)[10] have achieved remarkable results in NER tasks, demonstrating better performance than traditional methods. Consequently, deep learning-based methods have become the dominant solution in the NER domain. These methods utilize word-level embeddings[11–13], character-level embeddings[14–17], or composite embeddings[18–23] to learn the implicit features of words. Although these methods yield competitive results, they often overlook or oversimplify the learning of word- and character-level features from various perspectives.

Therefore, this paper proposes a multi-level feature embedding algorithm that simultaneously captures the multi-level semantic features of words from different perspectives. To more intuitively demonstrate these

[1]Guangzhou Institute of Science and Technology, Guangzhou 510540, China. [2]Nanfang College Guangzhou, Guangzhou 510970, China. ✉email: 1210437650@qq.com

multi-level features, we have provided an example sentence in Fig. 1, which showcases four types of word- and character-level features from both global and local perspectives: Global_Word, Local_Word, Global_Char, and Local_Char. These include frequently co-occurring lexical phrases (blue), keywords (green), capitalization (red), and suffixes (orange). The named entities in the output are displayed in black.

Inspired by attention mechanisms, Bi-directional Long Short-Term Memory (BiLSTM), and Convolutional Neural Network (CNN), we designed four parallel components—global word-level (GW), local word-level (LW), global character-level (GC), and local character-level (LC)—to simultaneously process the raw text and obtain word embeddings with more comprehensive semantic features.

However, a limitation of this model is that, while these features operate independently, they are interdependent and require a more effective integration of multi-level features. In recent years, feature fusion has shown outstanding performance in image processing[23,24] and has demonstrated great potential in the field of NER. In NER, feature fusion involves the integration of word-level and character-level features. This multi-level feature fusion strategy allows models to more accurately capture entity boundaries and understand the contextual dependencies of entities within the text.

However, existing studies on feature fusion[10,25–27] have predominantly adopted simple concatenation or weighted concatenation of different word representations, such as word-level features, character-level features, or other features, during the fusion process without considering the interdependencies between different features, which may lead to information omission. Therefore, we propose a feature fusion strategy based on cross-attention, which utilizes attention mechanisms to compute attention scores between word-level and character-level features and constructs interactive communication bridges between different features to capture their interdependencies. The fused features are then fed into the sequence-labeling layer to predict word labels.

In summary, the principal contributions of this paper are as follows:

- We propose an Attention-based Interactive Multi-Level Feature Fusion (AIMFF) framework for NER that integrates multi-level features from both word-level and character-level perspectives.
- We designed global word-level, local word-level, global character-level, and local character-level feature extraction components to capture the features from the perspectives of lexical phrases, keywords, capitalization, and suffixes, respectively.
- We propose a cross-attention fusion strategy that utilizes attention mechanisms to enable interactions between different features and captures the dependencies between multi-level features, thereby improving the performance of the NER model.
- Extensive experiments conducted on three NER datasets demonstrate that our proposed AIMFF outperforms a set of state-of-the-art baseline models.

The remainder of this paper is organized as follows. Section "Related work" reviews related work. Section "Our proposed framework" describes the proposed AIMFF. Section "Experiments" presents the experimental results and analyses. Section Conclusion presents conclusions and suggestions for future work.

Related work

Existing NER methods can be roughly divided into four categories: dictionary-based, rule-based, machine learning-based, and deep learning-based methods.

Dictionary-based methods

Early NER models often employed dictionary-matching methods, which primarily rely on the construction of dictionaries. In this method, a dictionary containing a set of predefined named entities is built and used for lookups during text parsing. For example, Savova et al.[28] proposed a dictionary lookup algorithm in which each named entity is mapped to a term. They constructed dictionaries based on terms from different datasets, which were then used to detect named entities from text. However, fully listing all relevant entities in a dictionary is a challenging task, greatly limiting the practical effectiveness and scalability of these methods.

Rule-based methods

Rule-based methods require experts to manually construct numerous rules for detecting specific entities in text. For example, Zhang and Elhadad[29] designed rules based on lexical syntactic patterns to recognize biomedical entities in clinical and biomedical texts. Etzioni et al.[30] devised rules based on a gazetteer to extract unsupervised named entities from the web. However, a limitation of these methods is their lack of generality or generalization, as most are only applicable to specific domains. Therefore, rule-based methods are often difficult to apply in mainstream applications.

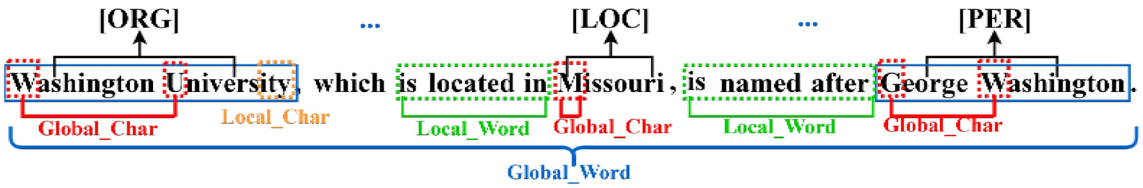

Fig. 1. The example of multi-level features.

Machine learning-based methods

Machine learning-based methods redefine the NER task as a multiclass sequence-labeling problem, primarily utilizing supervised learning. That is, they learn features from labeled text and then apply these features to process unlabeled text. Several machine learning algorithms have achieved satisfactory results for NER tasks. For example, Makino et al.[31] developed manual features from parts of speech and word patterns, integrated them with features generated by a Hidden Markov Model (HMM), and used a Support Vector Machine (SVM) as a classifier to produce entity recognition outcomes. Krishnan and Manning[32] employed a dual Conditional Random Fields (CRFs) method for entity recognition, where the initial CRF extracts local features and the subsequent CRF makes the final prediction using the features derived from the first CRF. Yanxiang and Chuwei[33] combined predefined rules with CRF to demonstrate the effectiveness of this method in geographical entity recognition. However, these methods heavily depend on the accuracy of feature engineering and require significant manual intervention and expert experience. Therefore, to reduce reliance on manual intervention in feature engineering, some researchers have proposed deep learning-based methods to achieve automatic feature learning.

Deep learning-based methods

Deep learning-based NER models have achieved impressive results due to advancements in deep learning. Compared with traditional methods, deep learning techniques have inherent advantages in automatically discovering hidden features from text data. Existing deep learning-based NER methods rely mainly on word, character, and composite embeddings for entity recognition, as described below. The initial NER technique, which leveraged word embeddings[12], utilized a CNN to extract local features and integrated a CRF layer for entity label prediction. To accommodate long-range dependencies, a subsequent investigation[11] substituted a CNN with a BiLSTM to enhance its ability to capture global features effectively. In a separate investigation[13], a fusion of a CNN and a BiLSTM was employed to boost the efficacy of language sequence labeling. Nonetheless, these methods overlook character-level features, possibly resulting in the omission of crucial information.

Character-level embeddings can be used to derive word representations because words can be regarded as a series of characters. Reference[15] introduced a CNN to extract local subword information at the character level, followed by an LSTM to process contextual features, and finally utilized the softmax function for prediction, thereby emphasizing the importance of character-level embeddings in NER. A different research endeavor[14] utilized a recurrent neural network to extract the character-level features of words from their context and improved NER using a CRF. In addition, character-level embeddings have demonstrated superior performance in handling various languages. For example, Dong et al.[16] achieved Chinese named entity recognition by using character-level embedding with radical Chinese features. Pham and Le-Hong[17] proposed an end-to-end recursive neural network model for Vietnamese named entity recognition and compared the performance of character- and word-level embeddings in their model. The experimental results demonstrated a 3.5% improvement in the F1-score with character-level embeddings compared to word-level embeddings. However, these methods suffer from information-omission issues.

Recent studies have demonstrated that various combinations of embeddings can optimize the performance of NER models. For example, Chiu and Nichols[34] proposed a new neural network architecture that utilizes BiLSTM and a CNN to capture both word- and character-level features and encode partial lexicon matches within the neural network. Lample et al.[35] used BiLSTM to extract character-level features and combined them with pretrained word embeddings to enhance the performance of NER models. Akbik et al.[18] proposed leveraging a trained character language model to generate a new character-level embedding called Contextual String Embedding (CS Embedding), in which pretrained word embeddings are stacked to add potential word-level semantics. Yoon et al.[19] proposed concatenating character-level embeddings with word-level embeddings trained on biomedical corpora as inputs to a BiLSTM-CRF model for sequence labeling, thereby improving the performance of the NER models. Cho et al.[21] proposed an NER model that enhances the representation of biomedical terms by combining two different character-level features extracted from a CNN and BiLSTM. In addition, some studies have incorporated other types of embedding. For example, Beltagy et al.[20] proposed a BERT-based method called SciBERT, which integrates token, segment, and positional embeddings into NER models. Yadav et al.[22] augmented word- and character-level embeddings using additional auxiliary affix information to enhance the performance of NER models. While these methods have all improved NER, few have considered multi-level features in NER from different perspectives, and thus do not fully utilize the available information.

However, these pieces of information are typically acquired using different models that operate independently. Simply combining them cannot capture the dependencies between different embeddings, thus failing to achieve an effective feature fusion. Feature fusion plays a crucial role in the field of image processing, significantly enhancing the accuracy and robustness of image analysis by integrating feature information from different sources or levels. for example, Dong et al.[24] proposed a feature fusion method based on the Spiking Neural Convolutional Network to address the loss of edge detail information in edge detection methods based on deep learning. Xian et al.[23] introduced a new framework called Multi-view Information Integration and Propagation (MVI^2P) to tackle the problem of person re-identification (Re-ID) under occlusion conditions. This framework is capable of integrating information from multiple viewpoints to enhance the feature representation of occluded target pedestrians. In the meantime, some researchers[10,25–27] have introduced feature fusion into the field of NER, achieving significant effects. For example, Wang et al.[10] proposed a cross-document attention module for merging representations of the same token across different documents. Yang et al.[25] designed four sets of word- and character-level features using attention mechanisms and adopted a concatenation strategy to adaptively fuse these features. Guan and Zhou[26] introduced a new word pair representation and combined it with randomly initialized distances and positional embeddings to enhance sentence representations. They then

utilized a simple convolution operation to capture the interaction information between different word pairs. Yang et al.[27] improved the performance of NER by introducing a document-level contextual feature based on previous research[25]. However, these methods employ strategies of concatenation or weighted concatenation for feature fusion, but fail to effectively capture the dependencies between different features.

In terms of multi-level feature extraction, the study most similar to ours is that of Yang et al.[25]. However, they focused on introducing a simple pooling operation after self attention-based word embedding in local word (LW) feature extraction, which overlooked the influence of the context before and after the word. By contrast, we utilize the convolutional operation of the CNN model, covering the contextual region before and after words with filters of certain widths, thereby extracting the contextual features of words and consequently enhancing the learning of LW features. In addition, our model introduces a cross-attention mechanism in feature fusion that can fully capture the dependency relationships between different features, thereby promoting the effective fusion of word and character embeddings. This contributes significantly to improving the performance of the NER model.

Our proposed framework

In this section, we present a detailed description of the proposed AIMFF method. We begin by demonstrating the overall architecture of AIMFF. Following that, we provide a detailed description of each layer within AIMFF.

Overall architecture of the proposed model

The overall structure of the proposed model is shown in Fig. 2. First, in the input layer, each word in the sentence is converted into both original word embeddings and character embeddings. The word embeddings are derived from a pretrained word-vector model, specifically using GloVe, whereas the character embeddings represent words by dividing them into a series of characters, each represented by a one-hot vector. Second, the original word and character embeddings are input into the feature extraction layer to extract the GW, LW, GC, and LC features of the words. These features are then input into the feature fusion layer for fusion. Finally, the fused features are input into the sequence-labeling layer to predict the labels of the words.

Input layer

In this section, we introduce two word representation methods. The first method matches words in a sentence using a pretrained word-vector lookup Table. In this study, we use the GloVe[36] model to train the word embeddings. For a given sentence S, the embedding of the t-th word w_t is obtained as follows:

$$x_t^w = e^w(w_t) \tag{1}$$

where e^w represents the pretrained word vector lookup table.

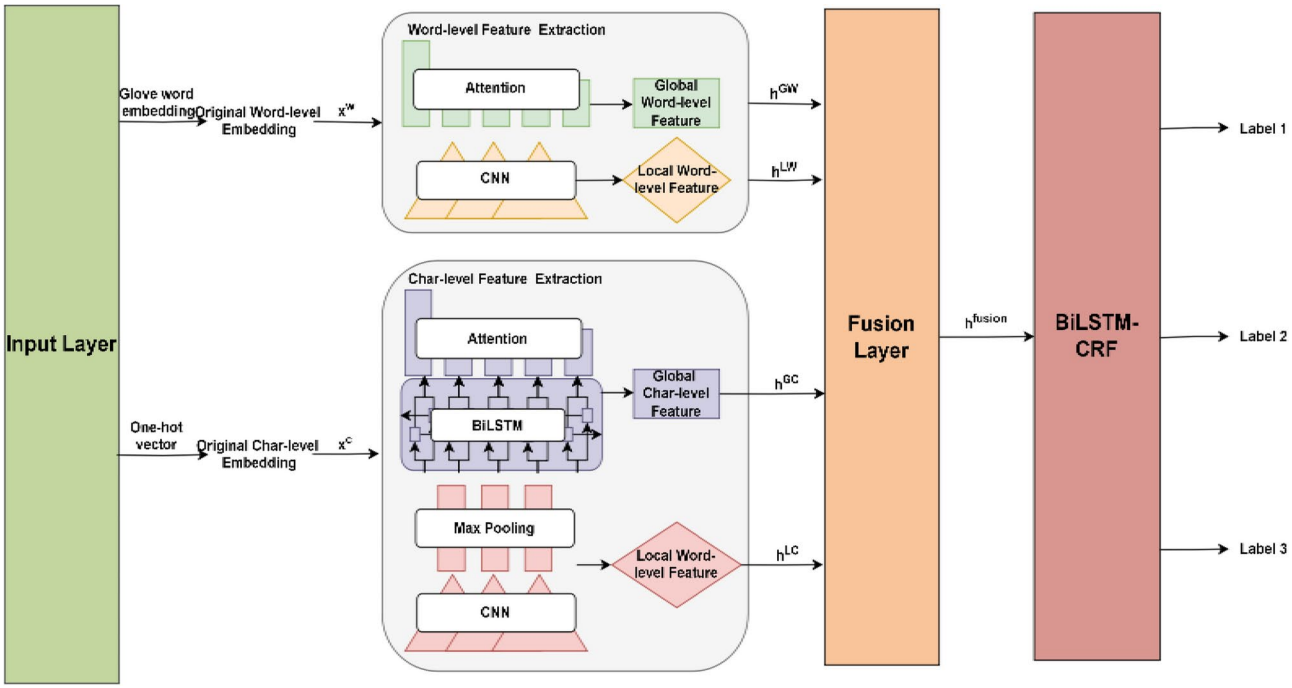

Fig. 2. Overall architecture of the proposed model.

The second method divides each word in a sentence into a series of characters, and then represents these characters using one-hot vectors. The embedding representation of the j-th character in the t-th word is as follows:

$$x_{tj}^{c} = e^{c}(c_j) \tag{2}$$

where e^c represents the character one-hot vector lookup table.

Feature extraction layer

Word-level feature extraction

Word embeddings have been widely used in NER modeling and have yielded significant results. However, existing NER models that utilize word embeddings fail to capture both global and local word-level features, resulting in omitted information. Therefore, this study introduces two neural networks that capture word-level features from these different perspectives, as detailed in the following sections.

(1) GW feature extraction

We designed a GW feature-extraction component to capture the GW features of words, as shown in Fig. 3. This component utilizes a self-attention mechanism, in conjunction to the original word embeddings, to capture the global contextual relationships between words.

For the input sentence $S=\{w_1,w_2,\ldots,w_n\}$, we calculate the score between each target word w_t and the other words in the sentence to obtain the similarity between the words. We use the Euclidean distance as the scoring function to calculate the similarity between words; this scoring function is expressed as follows:

$$score(w_t, w_i) = w_a \left| w_t - w_i \right| \tag{3}$$

where w_a is a trainable weight matrix.

Then, we use the softmax function to normalize the scores, thus generating attention weights $\alpha_{t,i}$ conditioned on the target word.

$$\alpha_{t,i} = \frac{\exp(score(w_t, w_i))}{\sum_k \exp(score(w_t, w_k))} \tag{4}$$

Subsequently, we compute the weighted sum of all word embeddings in the sentence with the attention weights, generating a GW feature vector h_t^{GW} for each target word.

$$h_t^{GW} = \sum\nolimits_i \alpha_{t,i} x_i^w \tag{5}$$

(2) LW feature extraction

Although attention mechanisms have sufficient advantages in capturing the global contextual relationships between words, they still lack the ability to extract LW features. Existing NER models typically use pretrained word embeddings as LW features without considering the influence of the context before and after the word. To address this issue, we designed a LW feature extraction component that applies the convolutional operations of the CNN model in addition to the original word embeddings, to capture the local contextual relationships before and after the words, as shown in Fig. 4.

If the size of the filter is k and $x_{i:j}^w$ is the concatenated segment representation from x_i^w to x_j^w, the calculation process for the LW feature vector h_t^{LW} of the t-th word in the text sequence is as follows:

$$h_t^{LW} = \mathrm{Re}LU(W \cdot x_{t-\frac{k}{2}:t+\frac{k}{2}}^{w} + b) \tag{6}$$

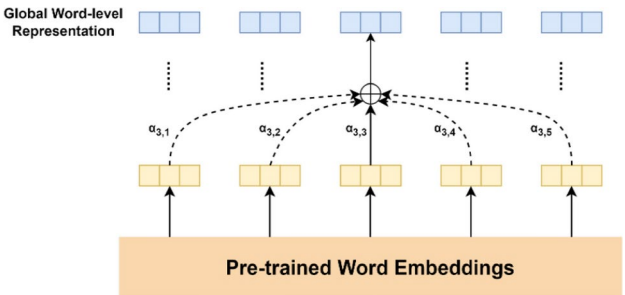

Fig. 3. The architecture of GW feature-extraction component.

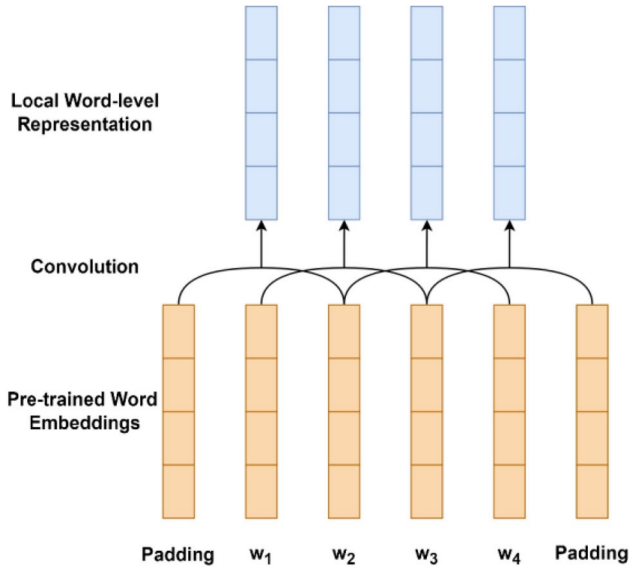

Fig. 4. The architecture of LW feature-extraction component.

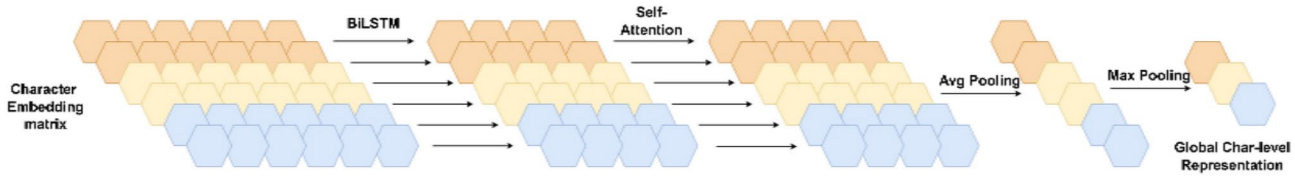

Fig. 5. The architecture of GC feature-extraction component.

where ReLU is an activation function, and W and b are the weight and bias parameters of the filter, respectively.

Character-level feature extraction

Recent studies have indicated that character-level features positively contribute to NER tasks. Introducing character-level embeddings into the embedding layers of NER models can significantly increase the number of processable words, thereby effectively avoiding higher out-of-vocabulary (OOV) rates. In addition, compared to word-level embeddings, character-level embeddings have fewer parameters and smaller dimensions, resulting in reduced computational costs. Therefore, the utilization of character-level embeddings can enhance the performance of NER models. To achieve this, we employed two neural network models to extract character-level embeddings.

(1) GC feature extraction

We designed a GC feature extraction component to capture the GC information of words, as shown in Fig. 5. This component utilizes a multi-head self-attention mechanism to extract more useful character-level information effectively.

The original character-level embeddings of the t-th word are fed into a BiLSTM to acquire the hidden layer states. The hidden layer states are expressed as follows:

$$h_i^{char} = \left[\overrightarrow{h}_i^{char} \oplus \overleftarrow{h}_i^{char} \right] \tag{7}$$

where $\vec{h}_i^{char}$ and $\overleftarrow{h}_i^{char}$ represent the forward output and backward output of the BiLSTM at time step t, respectively.

We introduced a self-attention mechanism after the BiLSTM network to capture the relationships between the target character and other characters within a word. Formally, similar to the GW feature extraction method, we take the character h_i^{char} as input and obtain the self-attention representation c_i as follows:

$$score(h_i^{char}, h_j^{char}) = w_b \left| h_i^{char} - h_j^{char} \right| \tag{8}$$

$$\alpha_{i,j} = \frac{\exp(score(h_i^{char}, h_j^{char}))}{\sum_k \exp(score(h_i^{char}, h_k^{char}))} \tag{9}$$

$$sar_{t,i} = \sum_k \alpha_{i,k} h_k^{char} \tag{10}$$

where w_b is a trainable weight matrix and $sar_{t,i}$ represents the self-attention representation of the i-th character of the t-th word. We sum the self-attention representation vectors of all the characters of the t-th word to form the self-attention representation sar_t of the t-th word.

We then utilized average pooling and max pooling to capture the GC information of the words.

For average pooling, we compute the average of the self-attention representations for each character. For max pooling, we select the maximum value among these averages as the GC representation of a word. Finally, we concatenate the multiple GC representations of the word to form the final GC feature vector, h_t^{GC}. The calculation process is shown in Eqs. (11) and (12).

$$head_k = MaxP(AvgP(sar_t^k)) \tag{11}$$

$$h_t^{GC} = Cat(head_1, head_2, ..., head_h) \tag{12}$$

where $head_k$ represents the representation of the t-th word in the k-th global character-level representation. AvgP stands for average pooling operation, MaxP stands for maximum pooling operation, and Cat represents concatenation operation.

(2) LC feature extraction

We designed a LC feature-extraction component to obtain character-level features from various perspectives, as shown in Fig. 6.

This component employs filters of multiple sizes to extract multigranular LC features of words. We use M sets of filters of different sizes to simultaneously perform sliding convolution operations on the character-embedding matrix of words, thereby obtaining different LC feature vectors within the words. These vectors are then concatenated to obtain the LC feature vector of the word, as shown in Eqs. (13) and (14) during the calculation process.

$$ConvMaxP_i = Cat(MaxP(Conv(W_c^i x_t^c + b_c^i))) \tag{13}$$

$$h_t^{LC} = Cat(ConvMaxP_1, ConvMaxP_2, ..., ConvMaxP_m) \tag{14}$$

where x_t^c represents the character-level feature matrix of the t-th word, Conv is the convolution operation, W_c^i and b_c^i are the convolutional kernel parameters and bias vectors of the i-th convolution operation, MaxP is the max-pooling operation, and Cat is the concatenation operation.

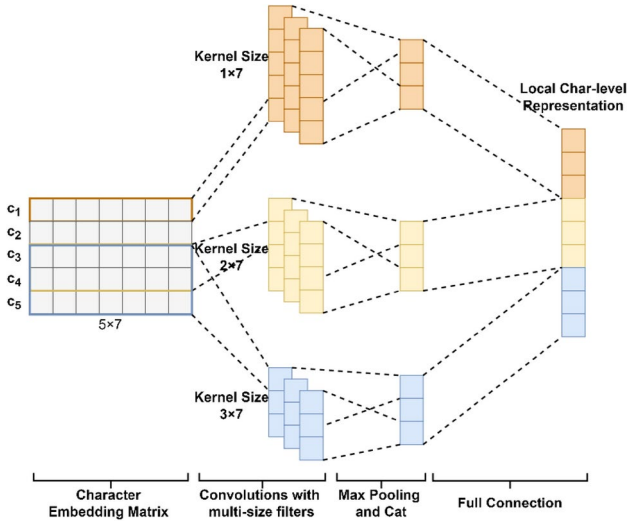

Fig. 6. The architecture of LC feature-extraction component.

Feature fusion layer

Multi-level feature fusion in NER tasks has been proven to be an effective and powerful strategy, enabling the extraction of pivotal information for achieving superior outcomes. We did not adopt the concatenation strategy[25] to fuse the multi-level features obtained from the above steps because this strategy ignores the dependencies between these features. Therefore, we introduce an interactive feature fusion strategy based on cross-attention to merge word- and character-level features and obtain embeddings with fused multi-level features. Furthermore, due to the differing dimensions of word- and character-level features, we use BiLSTM networks to standardize the dimensions of these features for interactive fusion. The specific computational process is illustrated in Eq. (15):

$$\begin{array}{ll} w' = BiLSTM[\mathrm{w}] & c' = BiLSTM[\mathrm{c}] \\ x_1 = w' \times c^T & x_2 = c' \times w'^T \\ y_1 = soft\max[x_1] & \mathrm{y}_2 = soft\max[x_2] \\ z_1 = y_1 \times w'^T & z_2 = \mathrm{y}_2 \times c'^T \\ o_1 = z_1 \cdot w' & o_2 = z_2 \cdot c' \\ Att = [o_1, o_2] & \end{array} \tag{15}$$

where w denotes a word-level feature, c denotes a character-level feature, and Att is the output of w and c after cross-fusion.

As shown in Fig. 7, we first employ a cross-attention mechanism to perform cross-fusion between the word- and character-level features. This generates four embedding vectors that contain both word- and character-level embeddings simultaneously. Second, these embedding vectors are split and recombined to form independent, complete word- or character-level embedding vectors.

Finally, we utilize a dimension reduction and combination module to form the final word-embedding h_t^{fusion}.

Sequence-labeling layer

The fused features of the words are fed into the sequence-labeling layer to predict their labels. BiLSTM can leverage all semantic information more comprehensively at higher levels. In addition, the CRF boosts the NER performance by considering neighboring labels, thus preventing mislabeling. For example, in a predicted label sequence annotated with BIO, the label following the B-ORG cannot be I-LOC or I-PER. During decoding, the Viterbi algorithm is used in the CRF to find the label sequence with the highest probability.

Formally, we assume that the output of the BiLSTM network is $\mathrm{x} = (\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_n)$, which is then inputted into the CRF module. For a predicted label sequence $\mathrm{y} = \{\mathrm{y}_1, \mathrm{y}_2, \ldots, \mathrm{y}_n\}$ of length n, s(x,y) denotes the scoring function for the input word sequence x with the predicted label sequence y, defined as follows:

$$s(x, y) = \sum_{i=0}^{n-1} T_{y_i, y_{i+1}} + \sum_{i=1}^{n} P_{i, y_i} \tag{16}$$

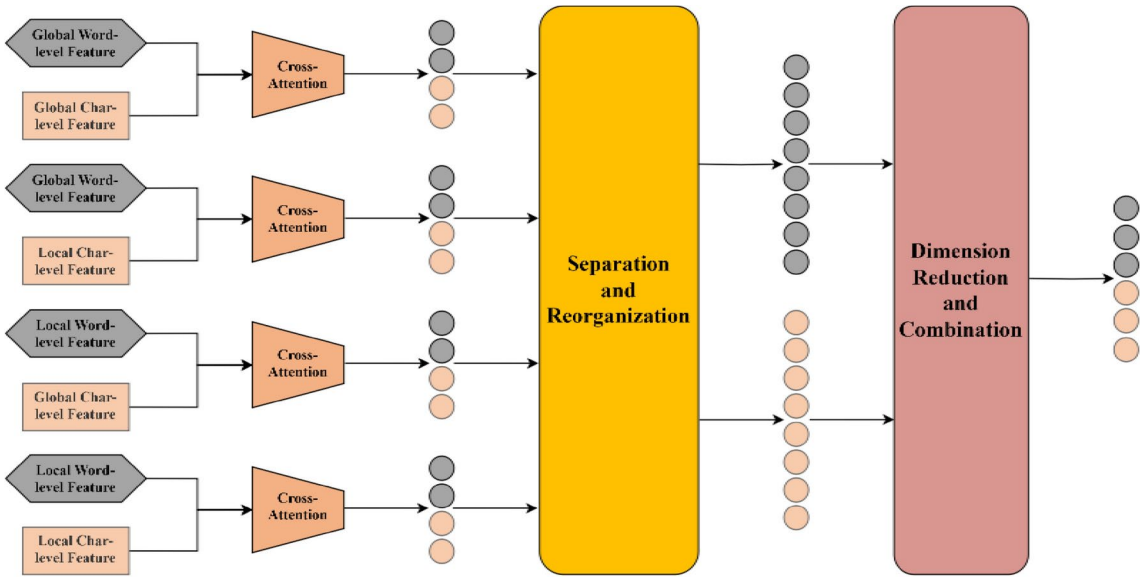

Fig. 7. Component of feature fusion.

where $T_{y_i,y_{i+1}}$ denotes the transition score from yi to yi + 1, and P_{i,y_i} denotes the score of the i-th word output by the BiLSTM network being labeled as the j-th tag.

Moreover, for all potentially predicted label sequences y_x, the CRF model further delineates the conditional probability p(y|x), as shown in Eq. (17).

$$p(y|x) = \frac{\exp(s(x,y))}{\sum_{\tilde{y}\in y_x} \exp(s(x,\tilde{y}))} \tag{17}$$

When the prediction is correct during the training phase, considering the maximum log probability is essential. In the decoding phase, it is necessary to identify the highest-scoring label sequence y* among all possible predicted label sequences.

$$y^* = \arg\max_{\tilde{y}\in y_x} s(x,\tilde{y}) \tag{18}$$

The specific implementation process of AIMFF is shown in Algorithm 1.

Input: Text sequence S={w1,w2,...,wn}
Output: Entity tag sequence **y**={y1,y2,...,yn}
1: **for** numbers of training iterations **do**
2: Original word-level embeddings $\mathbf{x}^w$←Equation (1)
3: Original char-level embeddings $\mathbf{x}^c$←Equation (2)
4: Global word-level feature: $\mathbf{h}^{GW}$←$f^{GW}(\mathbf{x}^w)$
5: Local word-level feature: $\mathbf{h}^{LW}$←$f^{LW}(\mathbf{x}^w)$
6: Global char-level feature: $\mathbf{h}^{GC}$←$f^{GC}(\mathbf{x}^c)$
7: Local char-level feature: $\mathbf{h}^{LC}$←$f^{LC}(\mathbf{x}^c)$
8: Fused feature $\mathbf{h}^{fusion}$ based on above features after cross fusion, segregation and reorganization, dimension reduction and combination operations
9: **y**←BiLSTM-CRF($\mathbf{h}^{fusion}$)
10: **end for**
11: **return** Entity tag sequence **y**

Algorithm 1. AIMFF for NER

Experiments

Datasets

Three publicly available benchmark datasets—namely, ConLL-2003, NCBI Disease, and JNLPBA—were employed to evaluate the performance of the model. ConLL-2003 consists of 20,744 sentences (15,562 for training, 1,729 for development, and 3,453 for testing) sourced from 1,393 English news articles. The NCBI-disease dataset includes 7,295 sentences from 793 abstracts (5,720 for training, 635 for development, and 960 for testing). JNLPBA comprises 24,806 sentences (18,491 for training, 2,055 for development, and 4,260 for testing) sourced from 2,400 abstracts in the MEDLINE database. Details of each dataset are provided in Table 1.

Experimental setup and evaluation metrics

We utilized pretrained word embeddings from GloVe[36] and initialized character embeddings as one-hot vectors as inputs. For the word embeddings, the number of dimensions was set to 128. For the character embeddings,

Dataset	ConLL-2003	NCBI-disease	JNLPBA
Target type	LOC,ORG,PER,MISC	SpecificDisease,CompositeMention, DiseaseClass,Modifier	DNA,RNA,Protein, cell_type,cell_line
Train	15,562	5720	18,491
Development	1729	635	2055
Test	3453	960	4260

Table 1. Statistical information of the three datasets.

the dimensionality was set to 84. In the LW feature extraction module, the lengths of the contexts before and after the word were set to 3. In the GC feature extraction component, the LSTM size was set to 64, and the number of self-attention heads was eight. In the LC feature extraction component, we employed three filter sizes (1, 2, and 3 characters), each containing 50 filters. We used the Adam optimizer[37] to iteratively update the model parameters. The learning rate was set to 1.5e-4, and the batch size was 8.

To prevent overfitting, we applied a dropout rate of 0.2 to the convolutional layers in the LW and LC components, the hidden layers of the BiLSTM network in the GC component, and the dimension reduction layer in the feature fusion network. We employed an early stopping strategy—stopping if the performance did not improve for four consecutive epochs—and repeated the experiments 10 times. All experiments were conducted on the same machine, using an Intel Core i7-9700 k CPU and an NVIDIA GeForce RTX 3080 Ti GPU. Several libraries, including Python, PyTorch, NumPy, and PyTorchCRF, were utilized to construct the model. We used three performance metrics—Precision, Recall, and F1-score—to evaluate the performance of the model.

$$precision = \frac{TP}{TP + FP} \tag{19}$$

$$recall = \frac{TP}{TP + FN} \tag{20}$$

$$F1 - score = \frac{2 * precision * recall}{precision + recall} \tag{21}$$

where TP represents the case in which the words of an entity are correctly predicted, FP represents the case in which words that are not entities are incorrectly predicted as such. Additionally, FN represents the case in which the predicted label for an entity does not match the true label, whereas TN represents the case in which words that are not entities are correctly identified as not being entities.

Baseline models

Our proposed model was compared with the following baseline models:

1) BiLSTM-CRF[11]**:** It employs BiLSTM to effectively learn features before and after the word and utilizes CRF to obtain dependencies among labels.

2) BiLSTM-CNNs[34]**:** It utilizes a CNN and pretrained word embeddings to extract word- and character-level embeddings, encoding partial vocabulary matches in the neural network.

3) NeuralNER[35]**:** It divides words into a series of characters and employs BiLSTM instead of a CNN to learn character-level features.

4) **CS Embeddings**[18]**:** It obtains context embeddings for all characters in the word; then, these embeddings are concatenated with pretrained word embeddings to form the final word representation.

5) SciBERT[20]**:** It introduces a BERT-based scientific text contextual embedding model that achieves state-of-the-art performance in multiple tasks.

6) CollaboNet[19]**:** It demonstrates that multiple single-task NER models can be employed to enhance prediction accuracy in the biomedical field.

7) AMFF[25]**:** It simultaneously obtains word embeddings at both the word and character levels from different perspectives and merges these word embeddings through an adaptive weighted concatenation strategy.

8) Att-MT-BLCC[10]**:** It aggregates representations of identical tokens across multiple documents and utilizes self-attention to learn the semantic relationships among them.

9) CAMFF[27]**:** It enhances the contextual extraction capability by incorporating document-level features in addition to AMFF[25].

10) PAMDFGA[26]**:** It introduces a new word-pair representation combined with randomly initialized distances and positional embeddings to enhance sentence representation. Subsequently, a simple convolution operation is used to capture the interaction information between different word pairs.

Comparison with previous methods

We compared our proposed model with ten previous methods on three public datasets—ConLL-2003, NCBI Disease, and JNLPBA—and the comparison results are shown in Table 2. From Table 2, it can be seen that, compared to some classical word-based and character-based NER models (BiLSTM-CRF[11], BiLSTM-CNNs[34], NeuralNER[35], CS Embedding [18], SciBERT[20], and CollaboNet[19]), our model achieved the highest F1-score values on all three datasets. This is mainly because these methods primarily use combined word- and character-level features for sequence labeling without employing effective feature fusion strategies, resulting in information omission. In addition, compared with classical NER methods, recent state-of-the-art methods based on feature fusion (AMFF[25], Att-MT-BLCC[10], CAMFF[27], and PAMDFGA[26]) achieved higher F1-score values, possibly because they capture richer semantic information through feature fusion strategies. However, our proposed model outperformed state-of-the-art feature fusion-based NER methods on all three datasets, with F1-score values of 94.75%, 94.05%, and 90.40%, validating its effectiveness. Specifically, our model improves the average F1-score performance on the three datasets by 9.4% compared to Att-MT-BLCC[10] and by 3.87% compared to PAMDFGA[26], primarily because these models use CNN and self-attention to enhance semantic relationships between different and identical words but fail to effectively integrate different features of words. Furthermore, our model surpasses AMFF[25] and CAMFF[27] on all three datasets, with average F1-scores exceeding those of AMFF[25] by 4.03% and CAMFF[27] by 3.45%, respectively, indicating better feature extraction capability of our

Model	ConLL-2003			NCBI Disease			JNLPBA		
	P(%)	R(%)	F(%)	P(%)	R(%)	F(%)	P(%)	R(%)	F(%)
BiLSTM-CRF[11]	92.78	87.43	90.02	85.47	74.32	79.51	73.47	68.27	70.77
BiLSTM-CNNs[34]	91.35	91.06	91.21	82.61	76.67	79.52	73.96	70.52	72.20
NeuralNER[35]	90.88	90.62	90.75	85.67	64.30	73.46	73.08	71.56	72.31
CS Embedding[18]	92.37	93.12	92.74	85.02	87.33	86.16	71.18	77.68	74.29
SciBERT[20]	88.46	89.13	88.79	84.32	89.06	86.63	70.73	80.36	75.24
CollaboNet[19]	87.31	81.47	84.29	80.50	81.42	80.95	72.92	82.42	77.38
AMFF[25]	94.83	94.12	94.48	89.60	94.76	92.11	79.09	81.99	80.51
Att-MT-BLCC[10]	90.74	90.21	90.47	84.48	80.52	82.45	–	–	–
CAMFF[27]	94.99	94.06	94.53	92.27	92.52	92.39	82.77	81.10	81.93
PAMDFGA[26]	–	–	–	89.76	91.35	90.55	–	–	–
AIMFF(ours)	94.96	94.57	**94.75**	93.94	94.57	**94.05**	90.99	90.19	**90.40**

Table 2. Comparison with previous methods. Significant values are in [bold].

Model	ConLL-2003	NCBI Disease	JNLPBA						
	P(%)	R(%)	F(%)	P(%)	R(%)	F(%)	P(%)	R(%)	F(%)
AIMFF_GW	89.01	89.18	90.32	88.72	87.90	87.29	88.79	89.17	88.52
AIMFF_LW	87.84	86.35	87.68	85.50	82.50	83.76	88.69	89.06	88.29
AIMFF_GC	83.48	86.97	84.58	85.47	83.17	82.13	65.42	80.88	72.34
AIMFF_LC	78.98	83.94	81.17	83.15	84.09	82.82	80.77	80.30	80.32
AIMFF_GW_LW	93.64	92.59	92.30	89.91	89.90	89.44	90.15	90.31	90.02
AIMFF_GW_LW_GC	93.34	92.43	92.60	92.50	94.01	92.69	90.57	90.17	90.11
AIMFF_GW_LW_GC_LC	**94.96**	**94.57**	**94.75**	**93.94**	**94.57**	**94.05**	**90.99**	**90.19**	**90.40**

Table 3. Ablation study results of the AIMFF framework. Significant values are in [bold].

fusion strategy for entities and stronger adaptability of the obtained fused features to datasets from different domains.

Additionally, we found that AIMFF performs significantly better when handling longer sentences. This is because our model is designed with a deep understanding of sentence structure, effectively capturing dependencies and semantic information within long sentences. For example, in the JNLPBA dataset, due to the characteristics of biomedical literature, sentences tend to be longer and contain complex structures, which allows our model to better leverage its strengths. This leads to the greatest improvement in F1-score for AIMFF on the JNLPBA dataset, with an increase of 8.47%. We also discovered that AIMFF has an advantage in dealing with domain-specific language, particularly in the NCBI-Disease and JNLPBA datasets. This is due to the incorporation of more refined word-level and character-level features in our model's embedding components, allowing the model to generate context-aware embeddings and better understand and process technical terms and domain-specific expressions, such as capitalization, abbreviations, and long words. We observed significant differences in the percentage improvement of the F1-score for AIMFF across different datasets. This phenomenon may be related to the complexity of the datasets and the adaptability of the model. For instance, on the ConLL-2003 dataset, due to the diversity of news articles and the generality of language, the improvement percentage of AIMFF's F1-score is relatively low, at only 0.22%, as other models in this field have already achieved high performance. In contrast, on the biomedical domain datasets NCBI-Disease and JNLPBA, the improvement percentage of AIMFF is higher, reaching 1.66% and 8.47%, respectively, possibly because our model better adapts to the specific needs of these fields. Through the analysis of dataset characteristics, we concluded that AIMFF performs better on certain datasets because it can effectively handle long sentences and domain-specific languages. These analyses not only provide deeper insights into our experimental results but also guide future research directions.

Ablation study

The proposed model utilizes four parallel components—GW, LW, GC, and LC—to capture the multi-level features of words and then integrates these features using the AIMFF strategy. GW and LW represent the two word-level feature extraction components, while GC and LC are the two character-level feature extraction components. Table 3 summarizes the results of the ablation study. Table 3 shows that integrating multiple components often results in greater competitiveness than integrating a single component. Specifically, for the NER model integrating a single component, the GW component contributes the most to the model, with F1-score values of 89.30, 87.29, and 88.52 on the three datasets, followed by the LW component, then the GC component, and finally the LC component, which contributes the least to the model. When two components are fused, the model's performance improves moderately. For example, on the ConLL-2003 dataset, its F1-score value is 92.30, which is an improvement of 2.19% and 5.27% compared to models integrating only the GW and

LW components, respectively. Additionally, for the NCBI Disease and JNLPBA datasets, the performance of the model combining the two components surpasses that of the models integrating a single component. When three components are fused, the model's performance is further enhanced, with the highest F1-score values for all datasets. As the number of integrated components increases, our proposed model demonstrates more effective and robust performance, mainly because the fusion module can eliminate redundant information and reinforce crucial information.

Furthermore, to demonstrate how the performance of AIMFF-GW_LW_GC_LC improves over using AIMFF-GW, AIMFF-LW, AIMFF-GC, or AIMFF-LC alone, we analyze the example in Fig. 1. AIMFF-GW can identify 'Washington University' as an organization because the word 'University' is typically associated with educational institutions. However, it may not distinguish whether 'Washington' refers to the university or the person 'George Washington'. AIMFF-LW can more precisely identify 'Missouri' as a location and 'George Washington' as a person by analyzing keywords such as 'is located in' and 'is named after'. However, local features may not be sufficient to recognize 'Washington University' as a complete entity. AIMFF-GC can identify these words as parts of named entities by recognizing the capitalization of all proper nouns such as 'Washington,' 'University,' 'Missouri,' and 'George Washington'. Yet, it may fail to understand the relationships between these entities. AIMFF-LC can recognize features within specific words, such as the '-ity' suffix following 'University,' indicating an organizational entity. However, it may not accurately recognize the multiple meanings of 'Washington' without global context. By integrating global word-level, local word-level, global character-level, and local character-level features, AIMFF-GW_LW_GC_LC can accurately identify 'Washington University' as an organization, understand 'Missouri' as a location, and 'George Washington' as a person. It can also recognize that 'Washington University' is named after 'George Washington,' demonstrating the relationships between entities. This multi-level feature fusion enables the model to comprehensively understand sentences, enhancing the accuracy and robustness of named entity recognition.

Parameter analysis

Four main parameters—filter number, LSTM size, head number, and filter width—were selected to analyze their impact on the performance of the model. The filter number represents the number of filters in the CNN model for the LC component. The LSTM size indicates the size of the hidden layer state in the BiLSTM model of the GC component. The head-number affects the feature extraction capability of multi-head self-attention in the GC component. The filter width represents the range of word context before and after the word in the LW component.

Impact of different filter numbers on the model

A CNN was added to the LC component, using three sets of filters of different sizes to extract character-level features. To validate the impact of different filter numbers in the LC component on the model, we conducted experiments on all three datasets, and the experimental results are shown in Fig. 8.

Figure 3 shows that in the ConLL-2003 dataset, the model achieves the highest F1-score when the number of filters is 50. In the NCBI Disease dataset, the model also achieves the highest F1-score when the number of filters is 50. In JNLPBA, the highest F1-score is achieved by the model when the filter number is 70. Overall, the model performs best when the filter number is 50. When the filter number exceeds 50, the performance of the model deteriorates, possibly because increasing the filter number indiscriminately leads to excessively large character embedding dimensions, increasing computational complexity and thus negatively impacting the model's performance.

Impact of different LSTM sizes on the model

We utilized a BiLSTM model in the GC component to extract character-level word embeddings. To validate the impact of different LSTM sizes in the BiLSTM model on the proposed model, we conducted experiments using the same three datasets. As the LSTM size increased, the F1-score remained relatively stable, as illustrated by the fluctuations in Fig. 9.

Figure 9 shows that in ConLL-2003, the model achieved the best performance with an LSTM size of 64, resulting in an F1-score of 92.75. Similarly, in NCBI-disease, the model reached its optimal performance with an LSTM size of 64. In the JNLPBA dataset, the model obtained the highest F1-score when the LSTM size was set

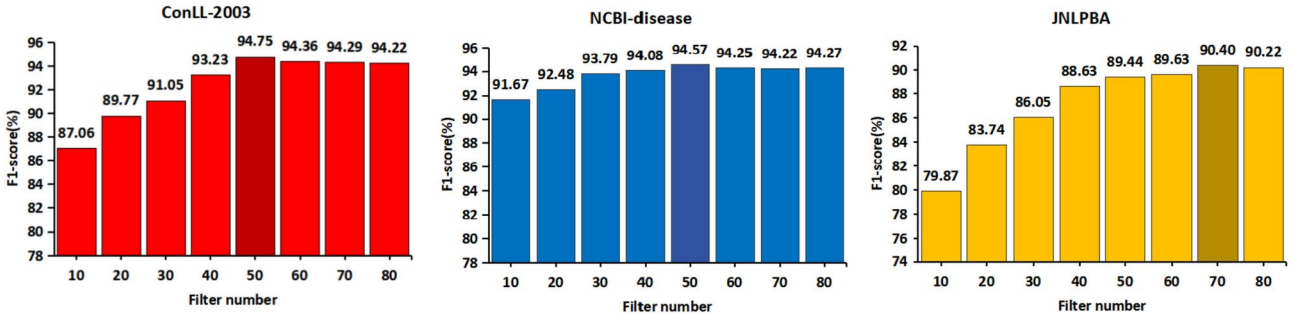

Fig. 8. Impact of different filter numbers on the model.

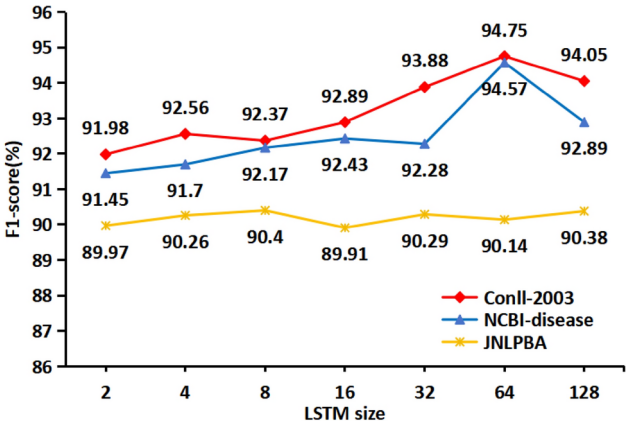

Fig. 9. Impact of different LSTM sizes on the model.

Head number	F1-score		
	ConLL-2003	NCBI-disease	JNLPBA
2	91.9	91.47	89.51
4	92.37	92.75	89.84
8	**94.75**	**94.57**	89.97
16	93.53	93.23	**90.40**
32	93.56	93.56	90.17
64	93.54	92.43	90.07
128	92.68	92.55	90.06

Table 4. Impact of different head numbers on the model. Significant values are in [bold].

to 64. Considering the performance of the LSTM size parameters across the different datasets, we selected 64 as the optimal parameter value for the LSTM size.

Impact of different head numbers on the model
We introduced multi-head self-attention in the GC component to enhance the character-level representation of words. Therefore, it is necessary to analyze the number of heads in the multi-head attention mechanism to determine the model's optimal performance. To validate the effect of different head numbers on the proposed model, we conducted comparative experiments on three datasets. The experimental results are listed in Table 4. From Table 4, it can be observed that the results varied as the number of heads increased sequentially. Note that the head number must be divisible by the dimension of the word embeddings. For ConLL-2003, the model achieved the highest F1-score when the number of heads was 16. For NCBI Disease, the best performance was observed when the head number was eight. Similarly, for JNLPBA, the F1-score was the highest when the head number was 16. However, the model's performance does not always improve with an increase in the number of heads, and as the number of heads exceeds 16, the performance of the model gradually declines. Overall, considering the performance across different datasets, we set the parameter value of the head number to 8 to balance model performance and computational efficiency.

Impact of different filter widths on the model
To explore the impact of the context before and after the word in the LW component, we conducted experiments on three datasets by varying the filter width parameter while keeping the other parameters fixed. The experimental results are presented in Fig. 10. From Fig. 10, it can be observed that different filter widths have varying effects on the model. In ConLL-2003, the model performed best when the filter width was three. For NCBI Disease and JNLPBA, the highest F1-scores were achieved when the filter widths were 5 and 3, respectively. Considering the performance across all datasets, we selected a filter width of 3 as the optimal parameter value. Overall, AIMFF maintains high performance as the filter width parameter varies. This indicates that the context features of a word play an important role in NER. Therefore, AIMFF demonstrates robustness and effectiveness in leveraging contextual information beyond the immediate context.

Impact of different feature fusion strategies on the model

To examine the effectiveness of the proposed fusion strategy, we conducted comparative experiments using two alternative fusion methods across the three datasets. The experimental results are presented in Table 5.

From Table 5, hGW, hLW, hGC, and hLC represent the global-word, local-word, global-character, and local-character embeddings, respectively. The direct concat strategy refers to the direct concatenation of these four

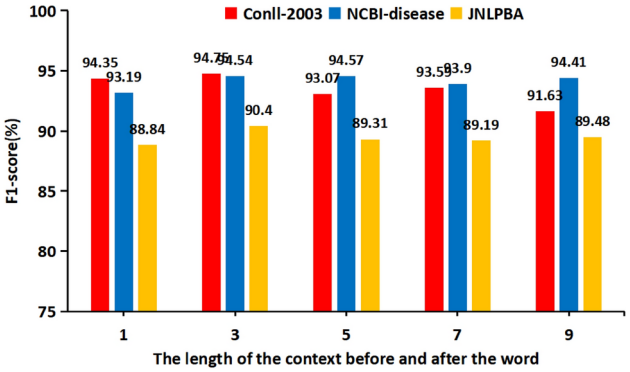

Fig. 10. Impact of different filter widths on the model.

Embeddings	Fusion Method	F1-score(%)		
		ConLL-2003	NCBI-disease	JNLPBA
h_{GW},h_{LW},h_{GC},h_{LC}	Direct Concat	91.70	91.08	86.94
	Weighted concat	93.48	93.14	89.76
	AIMFF	**94.75**	**94.05**	**90.40**

Table 5. Impact of different feature fusion strategies on the model. Significant values are in [bold].

embeddings. The weighted concat strategy concatenates the four embeddings using adaptive weighting. The proposed AIMFF strategy combines the four embeddings through cross-attention, as described in Section "Feature fusion layer". From Table 5, it can be observed that without using the AIMFF strategy, the performance of the model decreases notably. For example, in the NER model using the weighted concat strategy, its F1-score decreased by 1.52%, and in the NER model using the direct concat strategy, its performance decreased even further, reaching 3.06%. This indicates that our proposed AIMFF strategy can achieve better performance than other fusion strategies, validating its superiority and confirming the conclusions of Section "Comparison with previous methods".

Conclusion

We propose a novel attention-based interactive multi-level feature fusion framework called AIMFF, which effectively utilizes multi-level features of words to predict entity labels. The proposed framework first employs four parallel components—GW, LW, GC, and LC—to capture the corresponding features of words. We then fuse these features through a cross-attention mechanism to capture the dependencies between different features, thereby further enhancing NER performance. Moreover, extensive comparative experiments conducted on three datasets demonstrate that AIMFF outperforms various baseline models, obtaining state-of-the-art results. Finally, an impact analysis of different parameters and a comparative analysis of different fusion strategies indicate that the proposed AIMFF framework is effective and robust.

In future work, we plan to extend the proposed AIMFF framework by incorporating document-level features of words into the embedding layer, aiming to design a more comprehensive NER framework to further enhance the performance of NER models.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Received: 11 July 2024; Accepted: 13 January 2025
Published online: 24 January 2025

References

1. Shelar, H. et al. Named entity recognition approaches and their comparison for custom ner model. *Sci. Technol. Libr.* **39**(3), 324–337 (2020).
2. Banerjee, P. S. et al. A information retrieval based on question and answering and NER for unstructured information without using SQL. *Wirel. Person. Commun.* **108**, 1909–1931 (2019).
3. Roha, V. S., et al. Moo-cmds+ ner: Named entity recognition-based extractive comment-oriented multi-document summarization. European Conference on Information Retrieval. Cham: Springer Nature Switzerland (2023).
4. Mollá, D., Menno, V. Z. & Daniel, Smith. Named entity recognition for question answering. Australasian Language Technology Association Workshop. Australasian Language Technology Association (2006).
5. Li, Z. et al. Language model pre-training method in machine translation based on named entity recognition. *Int. J. Artif. Intell. Tools* **29**(7), 2040021 (2020).

6. Saeed, Z. et al. What's happening around the world? A survey and framework on event detection techniques on twitter. *J. Grid Comput.* **17**, 279–312 (2019).
7. Mykowiecka, A., Marciniak, M. & Kupść, A. Rule-based information extraction from patients' clinical data. *J. Biomed. Inform.* **42**(5), 923–936 (2009).
8. Kanya, N. & Ravi, T. Machine learning based biomedical named entity recognition. IET Chennai Fourth International Conference on Sustainable Energy and Intelligent Systems (SEISCON 2013). IET (2013).
9. Gasmi, H., Bouras, A. & Laval, J. LSTM recurrent neural networks for cybersecurity named entity recognition. *ICSEA* **11**, 2018 (2018).
10. Wang, D., Fan, H. & Liu, J. Learning with joint cross-document information via multi-task learning for named entity recognition. *Inf. Sci.* **579**, 454–467 (2021).
11. Huang, Z., Xu, W. & Yu, K. Bidirectional LSTM-CRF models for sequence tagging. https://arxiv.org/abs/1508.01991 (2015).
12. Collobert, R. et al. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.* **12**, 2493–2537 (2011).
13. Liu, L, et al. Empower sequence labeling with task-aware neural language model. Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 32. No. 1 (2018).
14. Kuru, O., Can, O. A. & Yuret, D. Charner: Character-level named entity recognition. Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers (2016).
15. Kim, Y., et al. Character-aware neural language models. Proceedings of the AAAI conference on artificial intelligence. Vol. 30. No. 1 (2016).
16. Dong, C., et al. Character-based LSTM-CRF with radical-level features for Chinese named entity recognition. Natural Language Understanding and Intelligent Applications: 5th CCF Conference on Natural Language Processing and Chinese Computing, NLPCC 2016, and 24th International Conference on Computer Processing of Oriental Languages, ICCPOL 2016, Kunming, China, December 2–6, 2016, Proceedings 24. Springer International Publishing (2016).
17. Character-Level, Vs. End-to-end recurrent neural network models for Vietnamese named entity recognition: word-level. Computational Linguistics: 15th International Conference of the Pacific Association for Computational Linguistics, PACLING 2017, Yangon, Myanmar, August 16–18, 2017, Revised Selected Papers. Vol. 781. Springer (2018).
18. Akbik, A., Blythe, D. & Vollgraf, R. Contextual string embeddings for sequence labeling. Proceedings of the 27th international conference on computational linguistics (2018).
19. Yoon, W. et al. Collabonet: Collaboration of deep neural networks for biomedical named entity recognition. *BMC Bioinf.* **20**, 55–65 (2019).
20. Beltagy, I., Lo, L., Cohan, A. SciBERT: A pretrained language model for scientific text. https://arxiv.org/abs/1903.10676 (2019).
21. Cho, M. et al. Combinatorial feature embedding based on CNN and LSTM for biomedical named entity recognition. *J. Biomed. Inf.* **103**, 103381 (2020).
22. Yadav, V., Sharp, R. & Bethard, S. Deep affix features improve neural named entity recognizers. Proceedings of the seventh joint conference on lexical and computational semantics (2018).
23. Xian, R. et al. Feature fusion method based on spiking neural convolutional network for edge detection. *Pattern Recognit.* **147**, 110112 (2024).
24. Dong, N. et al. Multi-view information integration and propagation for occluded person re-identification. *Inf. Fus.* **104**, 102201 (2024).
25. Yang, Z., et al. Attention-based multi-level feature fusion for named entity recognition. International joint conference on artificial intelligence (2020).
26. Guan, Z. & Zhou, X. A prefix and attention map discrimination fusion guided attention for biomedical named entity recognition. *BMC Bioinf.* **24**(1), 42 (2023).
27. Yang, Z., et al. Context-aware attentive multilevel feature fusion for named entity recognition. *IEEE Trans. Neural Netw. Learn. Syst.* (2022).
28. Savova, G. K. et al. Mayo clinical text analysis and knowledge extraction system (cTAKES): Architecture, component evaluation and applications. *J. Am. Med. Inf. Assoc.* **17**(5), 507–513 (2010).
29. Zhang, S. & Elhadad, N. Unsupervised biomedical named entity recognition: Experiments with clinical and biological texts. *J. Biomed. Inform.* **46**(6), 1088–1098 (2013).
30. Etzioni, O. et al. Unsupervised named-entity extraction from the web: An experimental study. *Artif. Intell.* **165**(1), 91–134 (2005).
31. Makino, T., Ohta, Y., Tsujii, J. Tuning support vector machines for biomedical named entity recognition. Proceedings of the ACL-02 workshop on Natural language processing in the biomedical domain (2002).
32. Krishnan, V. & Manning, C. D. An effective two-stage model for exploiting non-local dependencies in named entity recognition. Proceedings of the 21st international conference on computational linguistics and 44th annual meeting of the association for computational linguistics (2006).
33. He, Y., Luo, C. & Binyao, Hu. A geographic named entity recognition method based on the combination of CRF and rules. *Comput. Appl. Softw* **32**, 179–185 (2015).
34. Chiu, J. P. C. & Nichols, E. Named entity recognition with bidirectional LSTM-CNNs. *Trans. Assoc. Comput. Ling.* **4**, 357–370 (2016).
35. Lample, G., et al. Neural architectures for named entity recognition. https://arxiv.org/abs/1603.01360 (2016).
36. Pennington, J., Socher, R. & Manning, C. D. Glove: Global vectors for word representation. Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (2014).
37. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. https://arxiv.org/abs/1412.6980 (2014).

Author contributions

Yiwu Xu: Conceptualization, Methodology, Software, Writing–original draft. Yun Chen: Validation, Writing–review & editing, Supervision.

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to Y.C.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.