applied sciences

Article

NAS-CRE: Neural Architecture Search for Context-Based Relation Extraction

Rongen Yan [1,2], Dongmei Li [1,2], Yan Wu [3,4], Depeng Dang [5], Ye Tao [6] and Shaofei Wang [7,*]

[1] School of Information Science and Technology, Beijing Forestry University, Beijing 100083, China; reyan2023@bjfu.edu.cn (R.Y.); lidongmei@bjfu.edu.cn (D.L.)
[2] Engineering Research Center for Forestry-Oriented Intelligent Information Processing of National Forestry and Grassland Administration, Beijing 100083, China
[3] CSSC Systems Engineering Research Institute, Beijing 100094, China; aerolandx@gmail.com
[4] CSSC Intelligent Innovation Research Institute, Beijing 100094, China
[5] School of Artificial Intelligence, Beijing Normal University, Beijing 100875, China; ddepeng@bnu.edu.cn
[6] Zhongguancun Laboratory, Beijing 100094, China; taoye_402@buaa.edu.cn
[7] College of Information Engineering, Capital Normal University, Beijing 100048, China
* Correspondence: wangshaofei@cnu.edu.cn

Abstract: Relation extraction, a crucial task in natural language processing (NLP) for constructing knowledge graphs, entails extracting relational semantics between pairs of entities within a sentence. Given the intricacy of language, a single sentence often encompasses multiple entities that mutually influence one another. Recently, various iterations of recurrent neural networks (RNNs) have been introduced into relation extraction tasks, where the efficacy of neural network structures directly influences task performance. However, many neural networks necessitate manual determination of optimal parameters and network architectures, resulting in limited generalization capabilities for specific tasks. In this paper, we formally define the context-based relation extraction problem and propose a solution utilizing neural architecture search (NAS) to optimize RNN. Specifically, NAS employs an RNN controller to delineate an RNN cell, yielding an optimal structure to represent all relationships, thereby aiding in extracting relationships between target entities. Additionally, to enhance relation extraction performance, we leverage the XLNet pretrained model to comprehensively capture the semantic features of the sentence. Extensive experiments conducted on a real-world dataset containing words with multiple relationships demonstrate that our proposed method significantly enhances micro-F1 scores compared to state-of-the-art baselines.

Keywords: natural language processing; relation extraction; neural architecture search

Citation: Yan, R.; Li, D.; Wu, Y.; Dang, D.; Tao, Y.; Wang, S. NAS-CRE: Neural Architecture Search for Context-Based Relation Extraction. *Appl. Sci.* **2024**, *14*, 10960. https://doi.org/10.3390/app142310960

Academic Editor: Douglas O'Shaughnessy

Received: 30 October 2024
Revised: 21 November 2024
Accepted: 25 November 2024
Published: 26 November 2024

1. Introduction

Relation extraction is the task of extracting semantic relation information required by users from smaller-grained text sentences and is of essential research significance for question answering [1], reading comprehension [2], and machine translation [3]. In addition, the useful information extracted is an important component in the construction of a knowledge graph.

In recent years, researchers increasingly turn to deep learning (DL) techniques [4–6] and reinforcement learning (RL) methods [7] to improve feature extraction after one-hot encoding. For instance, Li et al. [8] utilize attention networks to tackle the challenge of sentence-level relation extraction. Sui et al. [9] propose a method for extracting multiple relations between the same entity pair within a single sentence, thereby enhancing the capability to capture complex relationships in text. However, these methods mainly focus on the extraction of relationships between entity pairs, overlooking the dependency relationships that other entity pairs within the sentence may have on the extraction of entities. To address this issue, Sorokin et al. [10] consider the influence of other relationship pairs within the sentence on entity extraction and demonstrate the effectiveness of their proposed method in the task of single-sentence relationship extraction. To simplify the expression, this paper defines the method as context-based relation extraction (CRE). Unlike traditional methods that focus solely on the entities themselves, the CRE method expands the scope to the entire sentence, allowing for a more comprehensive capture of the relationships between the target entities. Specifically, as shown in Figure 1, there exists a sentence containing two entity pairs (e_1, e_2) and (e_3, e_4). When extracting the relationship r_1 between e_1 and e_2, the influence of the relationship r_2 between the entity pair (e_3, e_4) is also considered.

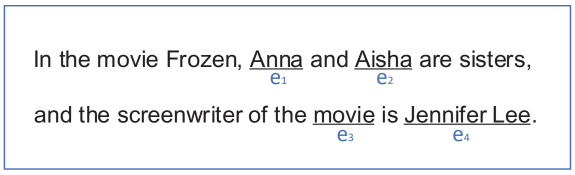
In the movie Frozen, Anna and Aisha are sisters,
e1
e2
and the screenwriter of the movie is Jennifer Lee.
e3
e4

Figure 1. When extracting the relationship between e_1 and e_2, the relationship between e_3 and e_4 in the sentence, which is related to the former relationship, is considered.

The focus of the aforementioned work has concentrated on innovations in model architecture and enhancements in performance, often neglecting the critical importance of internal parameter tuning, a process that typically relies on manual model design. This manual adjustment [11] not only consumes a significant amount of time and labor resources but is also susceptible to human factors, particularly the subjective biases and constraints of researchers' experiences. Taking sentence feature extraction as an example, although LSTM networks [12–14], as an advanced variant of RNNs, have been applied, parameter tuning still requires manual intervention. This demands considerable time investment and a deep level of expertise. Even with careful adjustments, the structure of LSTM networks may not reach an optimal configuration and often lacks adaptability to changes in data. Considering the complexity of deep learning in relationship extraction network models and the breadth of parameters involved, designing an efficient model is undoubtedly a challenging task. To fully explore and leverage feature information, researchers must meticulously adjust the network structure and parameters to achieve optimal performance. Therefore, the main research challenge in executing context-based relationship extraction tasks is how to implement automated searches for model structures and optimal configurations to efficiently complete the extraction tasks.

In response to the aforementioned challenges, this paper proposes a mechanism for the dynamic adjustment of the internal structure of neural networks, aiming to accurately capture relational expressions within sentences. This study adopts an encoder capable of precisely reflecting the semantic features of entire sentences. The proposed method is termed neural architecture search for context-based relation extraction (NAS-CRE), which seeks to effectively address the challenge of dynamic optimization of network structures. This work is inspired by the current wave of interest in neural architecture search (NAS). Within the NAS framework, we utilize an RNN controller to characterize RNN cells, with the goal of exploring the optimal RNN structure for extracting relational features between entity pairs within sentences. Additionally, to comprehensively extract semantic features from sentences, this study introduces the XLNet [15,16] pretrained model, which combines autoencoding (AE) and autoregressive (AR) language modeling. To the best of our knowledge, this work represents the first attempt to integrate relation extraction (CRE) with the optimal RNN structure obtained through NAS.

In summary, our contributions can be summarized as follows.

1. We define the concept of context-based relation extraction (CRE) and explore its unique attributes in the context of relation extraction tasks.
2. We propose a new paradigm, NAS-CRE, which serves as a method to investigate the potential effectiveness of integrating NAS and CRE. NAS-CRE utilizes an automatically optimized network architecture to extract relationships involving the target

entity, taking into account the dependencies of other entities in the context that are related to the target entity.
3. Extensive experiments on a large real-world database (https://github.com/UKPLab/emnlp2017-relation-extraction?utm_source=catalyzex.com (accessed on 10 August 2024)) show that our proposed method significantly and consistently outperforms baseline models.

The rest of this paper is structured as follows. In Section 2, related work is described. The overall description of our proposed method is presented in Section 3. The experimental description and experimental results are presented in Section 4. The discussion is showed in Sections 5 and 6 presents the conclusion and future outlook.

2. Related Work

Recently, relation extraction has become a research hotspot and is a key step in building knowledge graphs. In general, there are three mainstream solutions to the information extraction problem: supervised methods [17], unsupervised methods [18], and semisupervised [19] methods. Among them, supervised methods, which use a set of samples of known classes to train the model to meet the performance requirements, perform best.

Early existing methods used feature-based and kernel-based methods to extract relationships [20,21]. With the development of neural networks, researchers have tried to use different network structures to improve the performance of relation extraction. Zeng et al. and He et al. [22,23] proposed extracting vocabulary and sentence-level features using a convolutional neural network (CNN). Wang et al. and Li et al. [24,25] added attention to better identify ambiguities in sentences. CNN is too simple, but it made a groundbreaking breakthrough in relation extraction. Later, recurrent neural networks (RNNs) appeared based on researchers' perspectives. Vu et al. [26] concatenated a bidirectional RNN and used a ranking loss function to improve accuracy. Sorokin et al. [10,27,28] used a variant of RNN, the LSTM network, to achieve this task. As a result, RNN has become a classic method for solving relation extraction tasks. With the development of graph convolutional networks (GCNs), many researchers have begun to use improved graphs as feature extraction layers for information extraction. For example, Wang et al. and Sun et al. [29–31] used incrementally generated graphs to solve the problem of information extraction. Zhang et al. and Li et al. [32,33] added a graph convolutional network (GCN) to the syntax parse tree, which is conducive to relation extraction. However, the above work is based on the empirical design of the network structure, which artificially increases parameter constraints, such as specifying the use of CNN or a variant of RNN, LSTM. These designed networks ignore the impact of structural changes and may not be optimal. Therefore, these models cannot search for the optimal network structure and cannot accurately capture relationships within sentences.

Most of the above models directly use sentences to generate word vectors and require a large amount of data for training to prevent the model from overfitting. However, these data require human resources to label, which is not a good use of time. In this case, pretraining on many unlabeled corpora is particularly important. As a result, different pretrained models have begun to appear in the literature.

The pretrained model learns general semantic knowledge from a large corpus and migrates it to downstream tasks to improve low-resource tasks [16]. Pretraining enables the model to have generalization ability. Furthermore, a pretrained model is equivalent to a kind of regularization, which can prevent the model from overfitting downstream tasks [34]. After the first pretrained model appeared, embeddings from language models (ELMo) [35] and bidirectional encoder representations from transformers (BERT) [36,37] were proposed to generate word vectors. However, BERT [36], which uses a [MASK] in pretraining, also has shortcomings. For example, the use of a [MASK], which is an artificial symbol, is invalid in the actual data when fine-tuning is used and results in the inconsistency of fine-tuning in the pretraining step. Another disadvantage of using a [MASK] is that it is assumed that the predicted tokens are independent of each other, which results in unmasked tokens. For information extraction tasks that need to consider other relations in a sentence, independent mechanisms cannot relate multiple entity pairs in a sentence. To solve this problem, a new method to fuse AE and the AR language models, called XLNet, learns from the bidirectional context, avoiding the drawbacks brought by the mask method in the AE language model [38]. In our previous work [16], we demonstrated that using XLNet for named entity recognition tasks showed good performance. Going a step further, in this work, we use XLNet as an encoder for the information extraction task, producing vectors capable of expressing the semantic features of sentences.

NAS is one of the research hotspots in automatic machine learning. By designing cost-effective search methods, networks with strong generalization ability and user-friendly hardware requirements can be automatically acquired, which can significantly encourage the creativity of researchers. The significance of NAS is to solve the parameter adjustment problem of the deep learning model and obtain the optimal network. A structure search in deep learning is a direction that has received renewed attention in recent years. The most famous work in the structure search task used reinforcement learning to optimize the network structure [39], which started the increase in NAS research. The construction of a neural architecture search includes three aspects: search space, optimization algorithm, and model evaluation [40]. The earliest search method used in NAS was reinforcement learning. Reinforcement learning takes the network structure of each generation as an action, and the rewards of this action are represented by the evaluation results of this model. The difference in the representation of different reinforcement learning algorithms in NAS is how to design the search strategy of the agent. Zoph et al. [39] used RNN as a policy network to serialize and collect an encoded network structure. Currently, NAS is mainly used in the fields of image segmentation [41], graph matching [42], object detection [43], and pose estimation [44]. All of these applications are in the field of images; to the best of our knowledge, there are almost no uses of NAS in the field of natural language processing.

In contrast to previous endeavors, we advocate for the utilization of NAS to automatically search for and optimize RNN structures and parameter configurations. This approach obviates the need for extensive manual experimentation and adjustment, thereby facilitating the discovery of more efficient model architectures and driving advancements in the task of relation extraction. In addition, to improve the accuracy of relation extraction, we use the pretrained XLNet model to learn sentence representations from a bidirectional context.

3. Method

In this section, we describe the proposed NAS-RNN approach for the CRE problem. We first give the definition and formulation of the task. Then, the overall framework of the model is introduced. In particular, we provide details for the two major components in the framework, NAS-RNN and XLNet, for CRE.

3.1. Task Definition and Formulation

Given a sentence $s = \{x_1, x_2 ..., x_n\}$ containing at least two entity pairs $\{e_{h1}, e_{t1}\}$ $\{e_{h2}, e_{t2}\}$, where x_n represents the n-th words in sentence s, and e_{hm} and e_{tm} represent the m-th head entity and tail entity, respectively. CRE aims to first predict the relationship $\mathbf{r}$ of $\{e_{h1}, e_{t1}\}$ and then adopts it to help predict the relationship of $\{e_{h2}, e_{t2}\}$. For example, as illustrated in Figure 1, first, we predict the relationship of the entity pair $\{e_1, e_2\}$. When predicting the relationship of $\{e_3, e_4\}$, we connect the relationship code generated by $\{e_1, e_2\}$ and the relationship code generated by $\{e_3, e_4\}$ to form content encoding and finally extract the relationship of $\{e_3, e_4\}$ based on the content encoding.

3.2. Overall Framework

Figure 2 shows the overall framework of our proposed NAS-CRE. When dealing with a sentence, we use SentencePiece, which splits the sentence into many tokens. SentencePiece does not divide words according to spaces; instead, it divides English words into smaller semantic units and reduces the vocabulary. Each token in the sentence $s = \{x'_1, x'_2, ..., x'_n\}$

generates a 768-dimensional embedding vector X through the XLNet model pretrained on 13 GB of plain text [15]. In addition, each token has an encoded position P to mark the entity. For each word vector, we concatenate X and P.

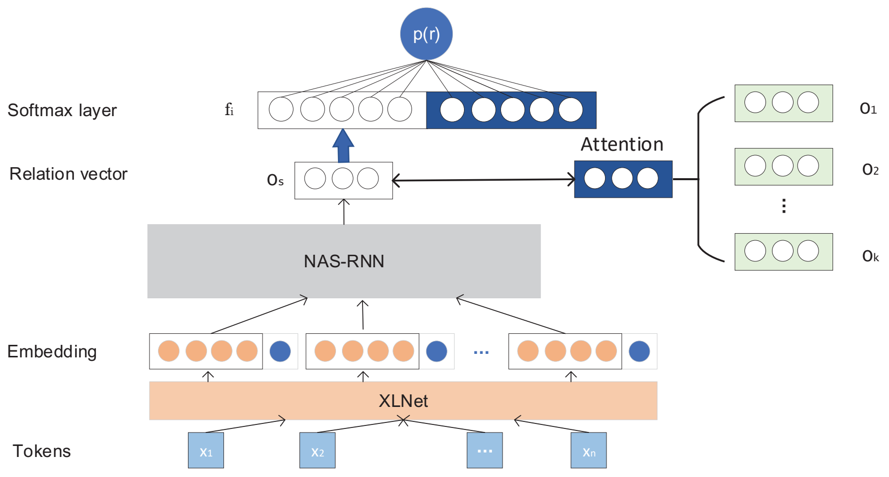

Figure 2. The overall structure of the neural architecture searches for context-based relation extraction. In the embedding layer, the blue part represents the splicing position encoder.

When we process the encoded position of entities in a sentence, we mark each word of the sentence. If the word is a head entity, we mark it as 2; if the word is a tail entity, we mark it as 3; and the rest of the words are marked as 1. To have a one-to-one correspondence between the encoded position and tokens, we mark the encoded position on the tokens generated by SentencePiece. Here, we give an example. Suppose we have the sentence "Willis lives in the USA". We want to extract the relationship of entity pair $\{Willis, USA\}$. The tokens generated by SentencePiece are ['_Will','_is','_lives','_in','_the', '_USA']. When performing position marking, '_Will' and '_is' are marked as 2, '_USA' is marked as 3, and the rest of the words are marked as 1. Therefore, the position is marked as [2, 2, 1, 1, 1, 3].

The token embedding and the encoded position are connected and fed to the NAS-RNN network to obtain the relationship o_s. For other entities in the sentence, we use the same method to obtain the relationship, o_1, o_2,..., o_k between entities, and then calculate their attention score o_s', i.e.,

$$o_s' = \sum_{i=0}^{k} \theta_i o_i \tag{1}$$

$$\theta_i = \frac{exp(score(o_i, o_s))}{\sum_{j=0}^{k} exp(score(o_j, o_s))} \tag{2}$$

where o_i represents each relationship, and $score(*, *)$ represents the score between entities calculated by the attention mechanism. We connect the obtained o_s with o_s', $\mathbf{o}$ = $[o_s,o_s']$. Then, to predict their relation, we feed the encoded relation $\mathbf{o}$ to the softmax layer, i.e.,

$$p(\mathbf{r} \mid \{e_1, e_2\}, s) = \frac{exp(a_i \cdot \mathbf{o} + b_i)}{\sum_{i=1}^{n_r} exp(a_i \cdot \mathbf{o} + b_i)} \tag{3}$$

where a_i is a weight, and b_i is a bias.

3.3. Sentence Representation

XLNet [15] is a pretrained model that trains the language model through many unlabeled language texts to obtain a set of model parameters and then uses these parameters to initialize the model. XLNet solves the abovementioned problems generated by BERT, which naturally introduces contextual information into the AR model and maintains consistency in the two stages of the AE model. The XLNet encoder uses the most advanced language model, transformer-XL, enabling XLNet to model longer dependencies. In addition, XLNet uses an AR language model and context words to predict the next word. The context here is limited to two directions, forward or backward. To add two-way semantic information, XLNet uses a permutation language model (PLM). PLM randomly generates an arrangement of a sentence, covers a certain amount of words at the end and finally uses AR to predict the words concealed in this arrangement. The AR method can better learn the dependencies between tokens, while the AE method can better utilize the deep bidirectional information.

As a result, XLNet makes full use of the context information of the sentence and considers the two-way information of the sentence to obtain an accurate representation of the sentence semantics. As shown in Figure 3, when the tokens in the sentence are put into XLNet, the order of the tokens in the sentence will be randomly shuffled. The next token is predicted in order from left to right. In other words, the prediction of the last token is based on the meaning of the preceding tokens.

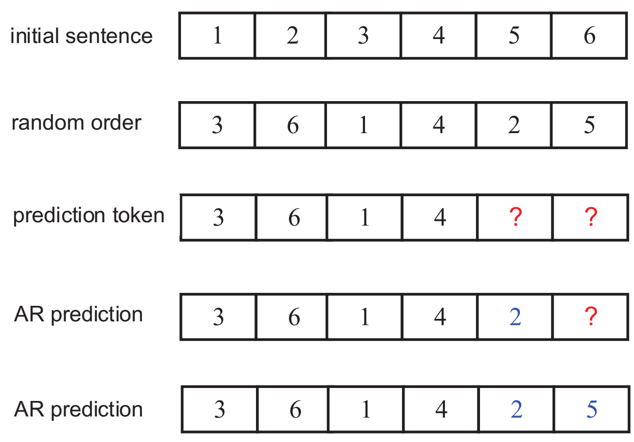

Figure 3. The principle of the permutation language model in XLNet. The number i represents the i-th token in a sentence. The blue text represents the knowledge predicted from the preceding context.

In this work, given a sentence $s = \{x_1, x_2 ... x_n\}$ and an entity pair, an XLNet model is used to produce vectors λ that fully characterize the semantic features of the sentence. To make the entities contain contextual semantic features, we filter the vectors of entities from the word vectors generated by the sentences. The model fine-tunes the task-specific basis of existing language models. After each token in sentence $s = \{x'_1, x'_2, ... x'_n\}$ is fed to XLNet, XLNet outputs word vectors λ_1 $\lambda_2 ...,\lambda_n$, where λ_i represents the i-th token.

3.4. Relational Feature Representation

A neural network can be regarded as a black box that can fit any function. As long as there is enough training data, given a specific x, the desired y can be obtained. Using these networks, only one input can be individually processed at a time, and the previous input and the next input are irrelevant. However, some tasks require information that can better handle sequences; that is, the previous input is related to the following input. For example, when we attempt to understand the meaning of a sentence, it is not sufficient to understand each word of the sentence in an isolated manner. Instead, we need to process the entire sequence of the connected words. In a sentence, the part-of-speech prediction of the previous word for the current word is very influential. To solve some similar problems and better process the information of the sequence, RNN is utilized.

RNN is one of the most commonly used models when dealing with time series problems using deep learning. There are many application fields of RNNs. As long as the problem of time sequence is considered, RNN can be used to solve it. The reason why RNN has excellent performance on time series data is that RNN will use the hidden node of the $t-1$ time slice as the input of the current time slice in the t time slice. The RNN structure is shown in Figure 4. In this figure, x_t represents the input information,

h_t represents the hidden state, and A represents the RNN structure. Using RNN, the information from the previous time slice is also used to calculate the content of the current time slice, and the current output of the sequence is relative to the previous output. The specific concept is that the network memorizes the previous information and applies it to the calculation of the current output, and the nodes between the hidden layers become connected. The input of the hidden layer includes not only the output of the input layer but also the output of the hidden layer at the previous moment.

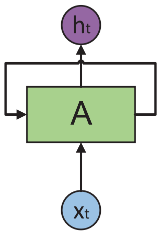

Figure 4. The structure of a recurrent neural network.

Although RNNs are widely used, they suffer from long-distance dependencies. The reason for the long-distance dependence is that after the nodes of the neural network have undergone many stages of computation, the features of the previous relatively long time slice have been covered. LSTM is a variant of RNN that has the ability to learn long-term and short-term information. The reason why LSTM can solve the long-term dependency problem of RNNs is because LSTM introduces a gate mechanism to control the flow and loss of features.

An RNN is a relatively mainstream network, but it is manually designed. Manually designing RNNs is a laborious and challenging task. NAS is a fully automatic neural structure search method that automatically learns applicable deep neural structures through algorithms [39]. When optimizing the network, NAS can use different search strategies, such as random search, Bayesian optimization, evolutionary algorithms, reinforcement learning, and gradient-based algorithms. In NAS-RNN, we use the common and effective reinforcement learning method to learn an RNN cell. NAS-RNN optimizes the network by maximizing the expected accuracy of the network on the validation set.

Compared to manually designed RNNs, NAS-RNN offers the following advantages:

1. Automatic Search: Traditional manual design methods often require extensive trial and error and significant human expertise. In contrast, NAS-RNN can automatically search for the optimal RNN architecture, alleviating the burden of manual design. By exploring various architectural spaces, it selects the structure best suited to the task, thereby enhancing the performance of the relation extraction model.
2. Efficient Parameters and Computation: Fixed-structure models in relation extraction tasks may possess excessive parameters, leading to high computational complexity. NAS-RNN, however, can identify more precise and compact structures tailored to specific task requirements. During the search process, NAS-RNN eliminates structural components that are irrelevant or perform poorly for the task, thereby reducing the presence of redundant parameters. Additionally, NAS-RNN can select and configure appropriate structural components, enabling the model to better capture the semantics and contextual information between relationships, thus improving overall performance.
3. Improved Generalization Capability: By automatically searching for the optimal structure, NAS-RNN can adaptively learn network architectures suitable for specific tasks. This enhances its modeling capability for complex and noisy sequential data, increasing the model's applicability and value in real-world scenarios.

NAS-RNN employs a controller composed of RNNs to randomly sample RNN structures that need optimization with a probability p. Sub-networks are trained on the training set, and accuracy R is calculated on the validation set. The accuracy R serves as a feedback signal, and NAS-RNN utilizes reinforcement learning as proposed by Zoph et al. [39]. The core idea of NAS-RNN is to learn to select RNN structures suitable for specific tasks through continuous trial and error. The parameters of the controller are continuously adjusted during the training process to generate better RNN structures, aiming to maximize performance metrics through reinforcement learning. This approach enables the automatic search and learning of RNN structures that are suitable for specific tasks, thereby enhancing the model's performance and generalization ability. Specifically, as shown in Figure 5, NAS-RNN uses the policy gradient algorithm to update the controller's parameters until model convergence. It learns to select appropriate RNN structures through interaction with the environment, maximizing a predefined performance metric via trial and error.

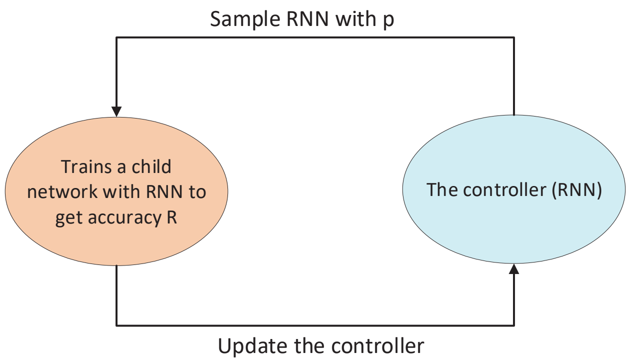

Figure 5. The flow chart of NAS-RNN, where the RNN in the left frame represents the optimized RNN.

To provide a detailed description of the NAS-RNN network, the following outlines its process.

Initialize the Controller Network and Target Network. First, initialize a controller network, which is a generative model used to produce action sequences for RNN network structures. The parameters of the controller network are denoted as χ_c. At the same time, initialize a target network, which is an RNN network to be optimized for training and evaluation on the training dataset. The initial parameters of the target network are θ_0.

Search and Training. In each iteration, use the current controller network and target network to generate a series of action sequences $c_{1:T}$. Convert each generated action sequence $c_{1:T}$ into the corresponding RNN network structure and train that network on the training dataset. Calculate the performance metric $R(c_{1:T};\theta)$ corresponding to each generated RNN network structure. Use gradient descent to update the parameters of the target network θ using the following formulas:

$$\mathrm{Loss}(D_{\mathrm{train}}, R(c_{1:T};\theta)) \tag{4}$$

$$\theta_{\mathrm{new}} = \theta_{\mathrm{old}} - lr \cdot \nabla \mathrm{Loss}(D_{\mathrm{train}}, R(c_{1:T};\theta)) \tag{5}$$

where D_{train} is the training dataset, θ_{new} is the updated parameter, θ_{old} is the current parameter value, lr is the learning rate, and $\nabla\mathrm{Loss}(\theta_{\mathrm{old}})$ is the gradient of the loss function with respect to the target network parameters θ_{old}.

Updating the Controller Network Parameters. Let the parameters of the controller network be denoted as χ_c. The objective function $J(\chi_c)$ can be defined as:

$$J(\chi_c) = E_{c_{1:T} \sim p(c_{1:T};\chi_c)}[R(c_{1:T};\theta)] \tag{6}$$

where $p(c_{1:T};\chi_c)$ represents the probability distribution of generating the action sequence $c_{1:T}$ given the controller network parameters χ_c. $R(c_{1:T};\theta)$ denotes the expected return after observing the action sequence $c_{1:T}$.

To update the controller network parameters χ_c, we can utilize the gradient ascent method, which requires calculating the gradient of the objective function $J(\chi_c)$ with respect to the parameters χ_c and updating the parameters accordingly.

Using the policy gradient method, we derive the gradient of the objective function $J(\chi_c)$ concerning the parameters χ_c as follows:

$$\nabla_{\chi_c} J(\chi_c) = E_{c_{1:T} \sim p(c_{1:T};\chi_c)}[\nabla_{\chi_c} \log p(c_{1:T};\chi_c) R(c_{1:T};\theta)] \tag{7}$$

where ∇_{χ_c} denotes the gradient with respect to the parameters χ_c, and $\log p(c_{1:T};\chi_c)$ is the logarithm of the probability of generating the action sequence $c_{1:T}$.

Finally, we can update the controller network parameters using the gradient ascent method, as shown in the following equation:

$$\chi_c^{t+1} = \chi_c^t + lr \nabla_{\chi_c} J(\chi_c^t) \tag{8}$$

where lr represents the learning rate, and χ_c^t denotes the parameter value of the controller network at iteration t, which is used to control the step size of the parameter update. By repeatedly executing the above steps, we continue until the convergence criteria are met or a preset number of iterations is reached.

Similar to the structure of LSTM units, the NAS-RNN structure also requires gating mechanisms. The NAS-RNN takes as input i_t, the previous hidden state h_{t-1}, and the cell state c_{t-1}. The outputs of the controller are the current hidden state h_t and the cell state c_t. The equations are as follows:

$$h_t = \text{sigmoid}(h_0^{new} \odot h_1) \tag{9}$$
$$h_0^{new} = \text{ReLU}(h_0 + c_{t-1}) \tag{10}$$
$$h_0 = \tanh(w_1 * i_t + w_2 * h_{t-1}) \tag{11}$$
$$h_1 = \text{ReLU}((w_3 * i_t) \odot (w_4 * h_{t-1})) \tag{12}$$
$$c_t = (w_3 * i_t) \odot (w_4 * h_{t-1}) \tag{13}$$

In these equations, w_1, w_2, w_3, and w_4 are weight parameters, while sigmoid, tanh, and ReLU are activation functions. From Equation (13), it can be observed that the cell state c_t in the network structure is dependent on i_t and h_{t-1}.

4. Experiments

The results from the experiments in this work are used to show that our proposed model can accurately represent the relationship of multiple entities in a sentence and can be used to apply the features of sentence representation and entity features to extract the relationship of entities in a sentence. To verify the performance of our proposed NAS-CRE method, we compare it with different baseline models. The experimental setup of the dataset and results are described in this section.

4.1. Dataset and Evaluation Metrics

To facilitate comparison with baselines, we conduct experiments on the dataset generated by [10]. Since we are performing the CRE task, we need to consider datasets with multiple relationships in a sentence, and we do not use datasets with only one relationship in a sentence. Such datasets are very rare, so we use the real data collected in [10]. Although we only used one dataset, this dataset is generated from Wikipedia data, which is a large dataset and satisfies the condition of containing multiple relations in one sentence. We believe that this dataset can be used in the demonstration of the effectiveness of the proposed method. The dataset, which was obtained from the English Wikipedia knowledge base, contains more than 28 million entities and 160 million relations. In each sentence in a full article, Ref. [10] extracted link annotations and retrieved the Wikidata entity ID corresponding to the linked article. There is an explicit one-to-one mapping between Wikidata entities and Wikipedia articles. This dataset has two types of annotations: entity and relation. In total, there are 353 different relationship types in this dataset. For the form and specific details of the dataset, we recommend readers refer to the original paper [10]. Table 1 shows the specific statistics of the dataset.

Following previous work [10], we evaluate the proposed model using the held dataset. We also use accuracy, micro-F1, and the macro-precision–recall curve to evaluate our model.

Table 1. Dataset statistics, where relation triple denotes the number of $\{entity_1, relation, entity_2\}$.

	Training	**Validation**	**Held**
Relation triple	284,295	113,852	287,902
Relation	578,199	190,160	600,804

4.2. Experimental Setup

In this study, the development process of NAS-CRE is meticulously carried out using the advanced integrated development environment PyCharm, ensuring efficient code writing and debugging. We chose Python 3.7 as the programming language, which provides a solid foundation for NAS-CRE implementation due to its powerful library support and simple syntax features. Additionally, the model construction relies on the industry-leading Keras deep learning framework, which is widely praised for its ease of use and flexibility. To further enhance computational performance, our server is equipped with an NVIDIA GeForce RTX series GPU. This high-performance graphics processing unit provides robust support for complex neural network training and inference tasks, ensuring efficient and precise execution of computational tasks.

4.2.1. XLNet

We used XLNet-Base to initialize the word vector with 12 hidden units, 768 hidden layers, and 12 heads. The dropout of XLNet was 0.1 by default. We also set summary_last_drop to 0.1. We used the trained XLNet model to encode sentences, resulting in word vectors that fit the semantic information of sentences. The learning rate was set to 0.0001.

4.2.2. Fine-Tuning

We used the TensorFlow framework uniformly to implement the proposed method and baselines. The model used the Adam [45] optimizer to calculate the gradient of the loss function and update the parameters. To save time, we set the number of RNN layers to 3. The hidden unit of NAS-RNN was 256. The remaining hyperparameters are listed in Table 2. We calculated the true positive and false positive of each relationship separately. Our model was trained on a Tesla V100 GPU for approximately 1 h.

Table 2. Hyperparameter statistics.

Hyperparameter	**Value**
Dropout_rate	0.5
Batch_size	128
Max_sentence_len	36
Learning_rate	0.001
RNN_units	256
Decay	0.0001
Position_embedding	3
Windows_size	3

4.3. Comparison Models

To verify the effectiveness of our proposed model, we used the CRE method in [10] as the baseline. We set the parameters according to the original text when reproducing the baseline to ensure a fair comparison. The baseline models are as follows:

LSTM-baseline [10] does not consider the impact of other entities of the sentence on the target entity. Instead, GloVe is used as an encoder, and LSTM is adopted to obtain the characteristics of the sentence.

ContextSum [10] considers the impact of other entities in the sentence on the target entity. Relationships between entities are represented using weighted sums. It is implemented using GloVe and LSTM.

ContextWeighted [10] considers the impact of other entities of the sentence on the target entity. Other relations in the sentence are spliced with attention scores during target entity extraction.

NAS-CRE is our proposed method and uses the XLNet pretraining model as an encoder. In addition, the RNN structure is automatically optimized to better consider the relationships of other entities in the sentence.

4.4. Results and Analysis

Table 3 shows the performance of different models on the hold-out dataset. Among them, the first set of data is the result achieved in our experimental environment based on the previous paper. LSTM-baseline, ContextSum, and ContextWeight are all encoded using GloVe to generate vectors. The second set of results is the method we proposed, using GloVe and XLNet as encoders. Specifically, the NAS-CRE model encoded with GloVe achieved a Micro-F1 score of 88.40 and an accuracy of 90.08%, while the NAS-CRE model encoded with XLNet achieved even better results, with a Micro-F1 score of 90.87 and an accuracy of 91.25%. From this table, we obtain the following observations.

1. The performance of our proposed method was consistently and significantly improved over all baselines. Specifically, when NAS-CRE was encoded with XLNet, the macro-F1 was approximately 7% better than that of ContextWeight. This demonstrates the robustness and adaptability of the proposed model, which uses NAS-RNN to automatically adjust the RNN structure.
2. The LSTM-baseline model had poor performance in the relation extraction task. Due to the lack of sentence content information, the impact of other entities in the sentence on the target entity was not considered. In contrast, the model that considers sentence content information was better, indicating that CRE had a significantly better effect when extracting sentence-level relations.
3. NAS-CRE performed better when encoded with XLNet than with GloVe, demonstrating the effectiveness of XLNet in fully understanding sentence semantics.

Table 3. Micro-F1 of different models on the held dataset.

Model	Encoder	Micro-F1	Accuracy
LSTM-baseline	GloVe	62.78	62.07
ContextSum	GloVe	76.51	76.70
ContextWeight	GloVe	82.36	85.25
NAS-CRE	GloVe	88.40	90.08
NAS-CRE	XLNet	90.87	91.25

To visualize the performance of various models, Figure 6 shows the results of the macro-precision–recall curve. It can be understood from the definition of precision and recall that the higher the precision and recall, the more efficient the model. In other words, the closer the drawn curve is to the upper right, the better the effect. As shown in Figure 6, we find that the models considering the content information of the sentence, such as our method, which is abbreviated as NAS-CRE and integrates NAS into the CRE task, are significantly better than LSTM baselines, indicating that the relationship between other entities in the sentence can be inferred from the relationship between entities. For the other two baseline methods, the effect of ContextWeight is better than that of ContextSum, and it can be noted that using the attention score to connect the relations between entities is better than simply using concatenation. In addition, we can see that NAS-CRE consistently and significantly outperforms all baselines by a recall of at least between 0.07 and 0.2.

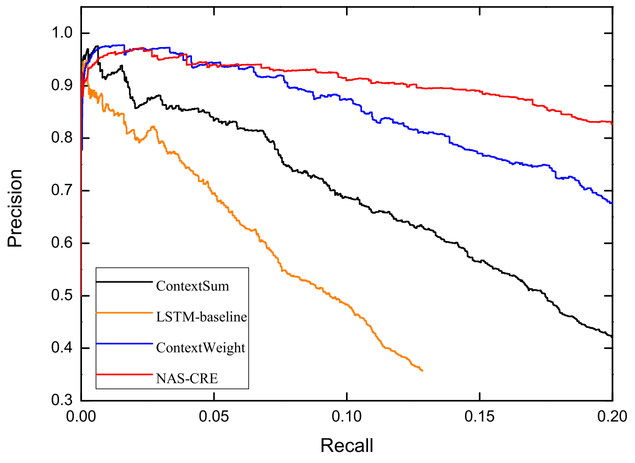

Figure 6. Aggregated macro precision–recall curves for different models.

4.5. Ablation Study

NAS-RNN and XLNet are two key components of our proposed model. To demonstrate the role these two components play in the overall model, we also conducted ablation experiments. XLNet is a generalized AR pretraining method based on the advantages and disadvantages of BERT. To verify the superiority of XLNet alone, we replaced XLnet with BERT and compared the micro-F1 value for information extraction. As shown in Table 4, NAS-CRE is the complete model we proposed, and 'w/o' is the abbreviation of 'without', which means to remove one or some modules. For example, 'w/o XLNet' means that XLNet was not used to generate word vectors. Instead, GloVe was utilized as the encoder. The abbreviation 'w/o NAS' represents the use of the XLNet pretrained model as the encoder and LSTM as the feature extractor, 'w/o NAS & XLNet' represents a model where GloVe and LSTM are used to complete information extraction, and 'BERT+NAS-RNN' represents the replacement of XLNet in our proposed method with the pretrained BERT model.

Table 4. The micro-F1 of the models after removing some components.

Models	**Micro-F1**	**Accuracy**
NAS-CRE	90.87	91.25
W/o XLNet	88.40	90.08
W/o NAS	86.32	87.27
W/o NAS & XLNet	82.36	85.25
BERT+NAS-RNN	89.46	90.52

As seen from Table 4, the difference between 'w/o XLNet' and the complete performance is not large, indicating that XLNet's role in the entire model is an attractive but inessential addition. However, the model without NAS performs much worse than the full model, indicating that after NAS automatically adjusts the structure of the RNN, the RNN can fully capture the features of the sentence. After removing both NAS and XLNet, the model is the same as the method proposed in the previous paper [10], and the effect is very poor. Furthermore, we can clearly observe in Figure 6 that NAS-CRE outperforms all baselines in almost 80% of the recalls. We find that when BERT is used instead of XLNet in our proposed model, the performance is only slightly worse than that of our proposed model but much better than that of the nonpretrained model. This shows that the pretraining model can fully express the meaning of the sentence in terms of sentence representation. In summary, in the whole framework, NAS-CRE and NAS have the most significant impact on performance, and XLNet also affects the model, but it is slightly inferior to NAS.

5. Discussion

The NAS-CRE model stands out in the field of relation extraction primarily due to its integration of cutting-edge NAS technology, which automates the fine-tuning of RNN structures to precisely match the specific requirements of the task. This significantly enhances the model's performance in complex relation extraction tasks. Additionally, the NAS-CRE model employs the advanced XLNet pre-trained model as its encoder, which demonstrates exceptional ability in decoding the deep semantic features of sentences. The self-attention mechanism of XLNet enables the model to capture long-range dependencies in text, which is crucial for accurately identifying complex relationships between entities. Compared to baseline models that focus only on the target entities, the NAS-CRE model comprehensively considers the potential influence of other entities in the sentence on the target entity. This makes the model not only more efficient but also more accurate when extracting cross-entity relationships.

When evaluating the time complexity of models, the time complexity of NAS-RNN is particularly crucial. Therefore, the following section will focus on comparing the time complexity of NAS-RNN with that of traditional RNN.

5.1. Time Complexity of RNN

The time complexity of an RNN mainly depends on the length of the input sequence T and the size of the hidden layer H. Suppose we have an input sequence of length T and the hidden layer size is H, the time complexity of the RNN can be broken down into the following parts:

Single-step computation complexity: At each time step t, the RNN needs to compute the hidden state h_t and possibly the output. The main computational cost comes from the matrix multiplication. Assuming the hidden layer size is H, and the input dimension is I, the computation complexity at each time step is $O(H \cdot (I + H))$, where H is the size of the hidden layer, and I is the input dimension.

Effect of sequence length: Since RNN processes the sequence step by step, the time complexity for an input sequence of length T is computed T times.

Therefore, the total time complexity of the RNN is:

$$O(T \cdot H \cdot (I + H)) \tag{14}$$

where T is the length of the input sequence, H is the size of the hidden layer, and I is the dimension of the input data.

5.2. Time Complexity of NAS-RNN

The time complexity of NAS-RNN consists of two main parts: architecture search and training of the optimal architecture.

5.2.1. Time Complexity of Architecture Search

A core feature of NAS-RNN is the automated architecture search. The architecture search aims to find the optimal network architecture using search strategies such as reinforcement learning.

Assuming that we perform N architecture evaluations, where each evaluation requires training a candidate architecture, the time complexity for training is the same as for a standard RNN, i.e., $O(T \cdot H \cdot (I + H))$, where: T is the sequence length, H is the size of the hidden layer, and I is the input dimension.

Thus, the time complexity of the architecture search is:

$$O(N \cdot T \cdot H \cdot (I + H)) \tag{15}$$

where N is the number of architecture evaluations. Since the architecture search usually requires exploring a large number of candidate architectures, N is often a very large number.

5.2.2. Time Complexity of Training the Optimal Architecture

Once the optimal architecture has been found through NAS, we can train it like a regular RNN. The training time complexity is the same as for traditional RNNs. Assuming the input sequence length is T, the hidden layer size is H, the input dimension is I, and the time complexity for training the optimal architecture is:

$$O(T \cdot H \cdot (I + H)) \tag{16}$$

Thus, the total time complexity of NAS-RNN is approximately:

$$O(N \cdot T \cdot H \cdot (I + H)) + O(T \cdot H \cdot (I + H)) \approx O(N \cdot T \cdot H \cdot (I + H)) \tag{17}$$

For NAS-RNN, the time complexity of the architecture search is $O(N \cdot T \cdot H \cdot (I + H))$, where N is the number of architecture evaluations. As a result, the total time complexity of NAS-RNN is usually larger than that of traditional RNNs. However, the advantage of NAS-RNN is that it can optimize the network architecture to achieve a more efficient and task-specific model compared to traditional RNNs.

6. Conclusions

In this paper, we propose a new framework for solving the CRE task that uses an RNN that automatically adjusts the network structure and parameters and considers the effect of other entities in the sentence on the target entity. Moreover, we adopt the XLNet pretrained model as an encoder to fully acquire semantic features between sentences. Experiments on the dataset generated from Wikipedia show that NAS-CRE fully captures the semantic features in sentences, maximizes the advantages of RNN, and achieves significant improvements over all baselines.

In future work, we will explore the following ideas:

- The effectiveness of NAS-CRE on the CRE task has been demonstrated. In the future, we will further explore the integration between NAS and other natural language tasks, such as document-level relation extraction and event extraction.
- In the CRE task, we use attention scores to integrate the relationships between entities in sentences. We will explore further using other methods to connect different relationships between sentences.
- We noticed that the training process of NAS-RNN is slightly long. We will further explore the internal mechanism of NAS, change the search strategy, and shorten the training time of NAS-RNN.

Author Contributions: Methodology, R.Y.; Validation, D.L.; Formal analysis, D.D.; Investigation, Y.W.; Data curation, S.W.; Writing—original draft, R.Y.; Writing—review & editing, R.Y.; Visualization, Y.T.; Funding acquisition, R.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by The Fundamental Research Funds for the Central Universities (BLX202356), National Nature Science Foundation of China under Grant (82374621).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The experimental code is accessible at https://github.com/dadadaray/NAS-CRE/.

Conflicts of Interest: The authors declare no conflicts of interest. The author, Yan Wu, was employed by the companies CSSC Systems Engineering Research Institute and CSSC Intelligent Innovation Research Institute. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Wang, Y.; Lipka, N.; Rossi, R.A.; Siu, A.; Zhang, R.; Derr, T. Knowledge graph prompting for multi-document question answering. In Proceedings of the AAAI Conference on Artificial Intelligence, Vancouver, BC, Canada, 20–27 February 2024; Volume 38, pp. 19206–19214.
2. Tong, X.; Yu, L.; Deacon, S.H. A Meta-Analysis of the Relation Between Syntactic Skills and Reading Comprehension: A Cross-Linguistic and Developmental Investigation. *Rev. Educ. Res.* **2024**, *2024*, 00346543241228185. [CrossRef]
3. Gui, S.; Shao, C.; Ma, Z.; Chen, Y.; Feng, Y. Non-autoregressive Machine Translation with Probabilistic Context-free Grammar. *Adv. Neural Inf. Process. Syst.* **2024**, *36*, 5598–5615.
4. Chen, H.; Wang, Y.; Guo, J.; Tao, D. Vanillanet: The power of minimalism in deep learning. *Adv. Neural Inf. Process. Syst.* **2024**, *36*, 7050–7064.
5. Li, X.; Li, Y.; Yang, J.; Liu, H.; Hu, P. A relation aware embedding mechanism for relation extraction. *Appl. Intell.* **2022**, *52*, 10022–10031. [CrossRef]
6. Zhang, M.; Qian, T.; Liu, B. Exploit Feature and Relation Hierarchy for Relation Extraction. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2022**, *30*, 917–930. [CrossRef]
7. Luo, F.M.; Xu, T.; Lai, H.; Chen, X.H.; Zhang, W.; Yu, Y. A survey on model-based reinforcement learning. *Sci. China Inf. Sci.* **2024**, *67*, 121101. [CrossRef]
8. Li, Z.; Hu, Z.; Luo, W.; Hu, X. SaberNet: Self-attention based effective relation network for few-shot learning. *Pattern Recognit.* **2023**, *133*, 109024. [CrossRef]
9. Sui, D.; Zeng, X.; Chen, Y.; Liu, K.; Zhao, J. Joint entity and relation extraction with set prediction networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, *35*, 12784–12795. [CrossRef]
10. Sorokin, D.; Gurevych, I. Context-aware representations for knowledge base relation extraction. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, 9–11 September 2017; pp. 1784–1789.
11. Yuan, L.; Cai, Y.; Wang, J.; Li, Q. Joint multimodal entity-relation extraction based on edge-enhanced graph alignment network and word-pair relation tagging. In Proceedings of the AAAI Conference on Artificial Intelligence, Washington, DC, USA, 7–14 February 2023; Volume 37, pp. 11051–11059.
12. Fabregat, H.; Duque, A.; Martinez-Romo, J.; Araujo, L. Negation-based transfer learning for improving biomedical Named Entity Recognition and Relation Extraction. *J. Biomed. Inform.* **2023**, *138*, 104279. [CrossRef]
13. Parsaeimehr, E.; Fartash, M.; Akbari Torkestani, J. Improving feature extraction using a hybrid of CNN and LSTM for entity identification. *Neural Process. Lett.* **2023**, *55*, 5979–5994. [CrossRef]
14. Sasibhooshan, R.; Kumaraswamy, S.; Sasidharan, S. Image caption generation using visual attention prediction and contextual spatial relation extraction. *J. Big Data* **2023**, *10*, 18. [CrossRef]
15. Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R.R.; Le, Q.V. Xlnet: Generalized autoregressive pretraining for language understanding. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 5753–5763.
16. Yan, R.; Jiang, X.; Dang, D. Named Entity Recognition by Using XLNet-BiLSTM-CRF. *Neural Process. Lett.* **2021**, *53*, 3339–3356. [CrossRef]
17. Zhang, S.; Wang, X.; Chen, Z.; Wang, L.; Xu, D.; Jia, Y. Survey of Supervised Joint Entity Relation Extraction Methods. *J. Front. Comput. Sci. Technol.* **2022**, *16*, 713.
18. Joshi, A.; Fidalgo, E.; Alegre, E.; Alaiz-Rodriguez, R. RankSum—An unsupervised extractive text summarization based on rank fusion. *Expert Syst. Appl.* **2022**, *200*, 116846. [CrossRef]
19. Zhang, B.; Zhang, H.; Le, V.H.; Moscato, P.; Zhang, A. Semi-supervised and unsupervised anomaly detection by mining numerical workflow relations from system logs. *Autom. Softw. Eng.* **2023**, *30*, 4. [CrossRef]
20. Sun, A.; Grishman, R.; Sekine, S. Semi-supervised relation extraction with large-scale word clustering. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Portland, OR, USA, 19–24 June 2011; pp. 521–529.
21. Nguyen, T.H.; Grishman, R. Employing word representations and regularization for domain adaptation of relation extraction. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, Baltimore, MD, USA, 23–24 June 2014; Volume 2, pp. 68–74.
22. Zeng, D.; Liu, K.; Chen, Y.; Zhao, J. Distant supervision for relation extraction via piecewise convolutional neural networks. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, 17–21 September 2015; pp. 1753–1762.
23. He, Z.; Chen, W.; Li, Z.; Zhang, W.; Shao, H.; Zhang, M. Syntax-aware entity representations for neural relation extraction. *Artif. Intell.* **2019**, *275*, 602–617. [CrossRef]
24. Wang, L.; Cao, Z.; De Melo, G.; Liu, Z. Relation classification via multi-level attention cnns. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Berlin, Germany, 7–12 August 2016; Volume 1, pp. 1298–1307.
25. Li, W.; Wang, Q.; Wu, J.; Yu, Z. Piecewise convolutional neural networks with position attention and similar bag attention for distant supervision relation extraction. *Appl. Intell.* **2022**, *52*, 4599–4609. [CrossRef]
26. Vu, N.T.; Adel, H.; Gupta, P.; Schütze, H. Combining recurrent and convolutional neural networks for relation classification. *arXiv* **2016**, arXiv:1605.07333.
27. Yang, D.; Wang, S.; Li, Z. Ensemble neural relation extraction with adaptive boosting. *arXiv* **2018**, arXiv:1801.09334.

28. Zhang, M.; Zhang, Y.; Fu, G. End-to-end neural relation extraction with global optimization. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, 9–11 September 2017; pp. 1730–1740.
29. Wang, S.; Zhang, Y.; Che, W.; Liu, T. Joint extraction of entities and relations based on a novel graph scheme. In Proceedings of the IJCAI, Stockholm, Sweden, 13–19 July 2018; pp. 4461–4467.
30. Wang, G.; Liu, S.; Wei, F. Weighted graph convolution over dependency trees for nontaxonomic relation extraction on public opinion information. *Appl. Intell.* **2022**, *52*, 3403–3417. [CrossRef]
31. Sun, Q.; Zhang, K.; Lv, L.; Li, X.; Huang, K.; Zhang, T. Joint extraction of entities and overlapping relations by improved graph convolutional networks. *Appl. Intell.* **2022**, *52*, 5212–5224. [CrossRef]
32. Zhang, Y.; Qi, P.; Manning, C.D. Graph convolution over pruned dependency trees improves relation extraction. *arXiv* **2018**, arXiv:1809.10185.
33. Li, Z.; Sun, Y.; Zhu, J.; Tang, S.; Zhang, C.; Ma, H. Improve relation extraction with dual attention-guided graph convolutional networks. *Neural Comput. Appl.* **2021**, *33*, 1773–1784. [CrossRef]
34. Sikaroudi, M.; Hosseini, M.; Gonzalez, R.; Rahnamayan, S.; Tizhoosh, H. Generalization of vision pre-trained models for histopathology. *Sci. Rep.* **2023**, *13*, 6065. [CrossRef]
35. Mishra, S. Multi-dataset-multi-task neural sequence tagging for information extraction from tweets. In Proceedings of the 30th ACM Conference on Hypertext and Social Media, Hof, Germany, 17–20 September 2019; pp. 283–284.
36. Mulyar, A.; Uzuner, O.; McInnes, B. MT-clinical BERT: Scaling clinical information extraction with multitask learning. *J. Am. Med. Inform. Assoc.* **2021**, *28*, 2108–2115. [CrossRef]
37. Zhang, N.; Ye, H.; Deng, S.; Tan, C.; Chen, M.; Huang, S.; Huang, F.; Chen, H. Contrastive information extraction with generative transformer. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2021**, *29*, 3077–3088. [CrossRef]
38. Han, X.; Zhang, Z.; Ding, N.; Gu, Y.; Liu, X.; Huo, Y.; Qiu, J.; Zhang, L.; Han, W.; Huang, M.; et al. Pre-trained models: Past, present and future. *AI Open* **2021**, *2*, 225–250. [CrossRef]
39. Zoph, B.; Le, Q.V. Neural architecture search with reinforcement learning. *arXiv* **2016**, arXiv:1611.01578.
40. Elsken, T.; Metzen, J.H.; Hutter, F. Neural architecture search: A survey. *J. Mach. Learn. Res.* **2019**, *20*, 1997–2017.
41. Xiao, H.; Li, L.; Liu, Q.; Zhu, X.; Zhang, Q. Transformers in medical image segmentation: A review. *Biomed. Signal Process. Control* **2023**, *84*, 104791. [CrossRef]
42. Mao, C.; Wu, Y.; Xu, J.; Yu, S.H. Random graph matching at Otter's threshold via counting chandeliers. In Proceedings of the 55th Annual ACM Symposium on Theory of Computing, Orlando, FL, USA, 20–23 June 2023; pp. 1345–1356.
43. Zou, Z.; Chen, K.; Shi, Z.; Guo, Y.; Ye, J. Object detection in 20 years: A survey. *Proc. IEEE* **2023**, *111*, 257–276. [CrossRef]
44. Liu, J.; Weng, Z.; Zhu, Y. Precise region semantics-assisted GAN for pose-guided person image generation. *CAAI Trans. Intell. Technol.* **2024**, *9*, 665–678. [CrossRef]
45. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2015**, arXiv:1412.6980.