SCL-SKG:Software Knowledge Triplet Extraction with Span-level Contrastive Learning

Mingjing Tang (tmj@ynnu.edu.cn)
Yunnan Normal University
Shu Zhang
Yunnan Normal University
Ming Zheng
Anhui Normal University
Zifei Ma
Yunnan Agriculture University
Wei Gao
Yunnan Normal University

Research Article

Keywords: Knowledge Graph, Software knowledge triplet, Entity extraction, Relation extraction, Contrastive learning, StackOverflow

Posted Date: October 24th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-2182156/v1

SCL-SKG:Software Knowledge Triplet Extraction with Span-level Contrastive Learning

Mingjing Tang[1,2], Shu Zhang[3], Ming Zheng[4], Zifei Ma[5], Wei Gao[3]

[1] Key Laboratory of Educational Informatization for Nationalities Ministry of Education, Yunnan Normal University, Kunming, China
[2] Yunnan Key Laboratory of Smart Education, Yunnan Normal University, Kunming, China
[3] School of Information Science and Technology, Yunnan Normal University, Kunming, China
[4] School of Computer and Information, Anhui Normal University, Wuhu, China
[5] School of Water Conservancy, Yunnan Agriculture University, Kunming, China

Abstract

The text of software knowledge community contains abundant knowledge of software engineering field. The software knowledge triplet can be extracted automatically and efficiently to form the software knowledge graph, which is helpful for software knowledge-centric intelligent applications, such as intelligent question answering, automatic document generation and software expert recommendation. Most existing methods are confronted with problems of task dependence and entity overlap. In this paper, we propose a software knowledge triplet extraction method based on span-level contrastive learning. From the level of sentence sequence modelling, we model the sentence sequence with span as a unit, and generate abundant positive and negative samples of entity span through the span representation layer to avoid the problem that the token-level method cannot select overlapping entities. From the level of feature learning, we propose supervised entity contrastive learning and relation contrastive learning, which obtain enhanced feature representation of entity span and entity pair through positive and negative sample enhancement and contrastive loss function construction. Experiments are conducted on the dataset which is constructed based on texts of the StackOverflow, and show that our approach achieves a better performance than baseline models.

Keywords: Knowledge Graph; Software knowledge triplet; Entity extraction; Relation extraction; Contrastive learning; StackOverflow

1. Introduction

In the context of big data software engineering, software knowledge communities (e.g., StackOverflow) generate massive community texts, which contain rich software domain knowledge such as software development activities, software development technology, software development tools and software project management [1]. Therefore, automatic and efficient extraction of software knowledge entities and their semantic relations from software knowledge community texts, and construction of software knowledge graph, are helpful for software knowledge-centric intelligent applications, such as intelligent question answering, automatic document generation and software expert recommendation, and play an important role in improving software development efficiency and software production quality.

Existing researches [2-8] are mostly based on pipeline method, and entity extraction and relation extraction are modeled as two independent tasks. The model implementation of these methods is relatively simple and flexible, but the interaction and association of entity extraction and relation extraction are ignored, causing many problems [9]. On the one hand, due to the error propagation of serial tasks, the quality of tasks in the previous stage directly affects the performance of tasks in the next stage. On the other hand, due to the independent modeling of entity extraction and relation extraction, the parameters and the information of the two tasks do not share or interact, semantic information and dependency information are lost, resulting in redundant entities and high error rate.

Different from the pipeline method, the joint learning method models entity extraction and relation extraction as a task, and uses the joint loss function to enhance the association and information sharing between tasks. The the overall performance of the model is improved and developed into the mainstream method of information extraction task [10][11].

According to the different modeling objects, the extraction of triples based on joint learning method can be divided into the method based on parameter sharing and the method based on sequence tagging. In order to solve the problem of manual feature extraction, the method based on parameter sharing [12][13][14] uses neural network to construct feature encoder and feature sharing layer to realize automatic feature extraction and model parameter sharing, which can alleviate the problems of error propagation and inter-task relation dependence. However, this kind of methods inherits the sequence relation of subtasks in essence, which will produce redundant entities with no matching relation and affect the quality of joint extraction. In order to solve the problem of redundant entities mentioned above, the method based on sequence tagging [15][16][17]uses joint tagging to label the entity location, entity relation type, entity role and so on, and transforms the joint learning model into a sequence tagging model to realize simultaneous extraction of entity and relation. However, such methods are usually token-level tasks, and the inherent sequential characteristic of sentence sequences will result in the inability to select overlapping entities, which will lead to the entity overlap problem in relation extraction tasks.

The text of software knowledge community is unstructured user-generated content, and the semantic relation of domain entities is complex. The traditional sequence modeling method is used to extract software knowledge triplet, which will cause problems of entity overlap and error propagation, and will affect the quality of software knowledge graph construction.

Motivated by above problems, this paper proposes a method based on span-level contrastive learning for software knowledge triplet extraction. From the perspective of modeling object, this method models the sentence sequence with span as a unit, which can avoid the disadvantages of token-level sentence sequence modeling and effectively alleviate the entity overlap problem. From the perspective of feature learning, this method introduces contrastive learning to obtain more distinct feature representation of entity span and entity pair, so as to improve the accuracy of classification prediction. The main contributions of this paper are as follows:

1) We propose a span-level contrastive learning model named SCL-SKG for software knowledge triplet extraction from software knowledge community texts. The proposed SCL-SKG takes the sentence sequence with span as a unit, and generates abundant positive and negative samples of entity span as data augmentation strategies for contrastive learning.
2) We introduce supervised entity contrastive learning algorithm and supervised relation contrastive learning algorithm to learn effective feature representation of entities and entity pairs that are more suitable for downstream classification tasks.
3) The experimental results showed that the proposed model achieves better performance than the benchmark models, which demonstrates its effectiveness.

The remainder of this paper is organized as follows. Related works are reviewed in Section 2, and Section 3 presents each module of the proposed approach in detail. Section 4 presents the details of benchmark dataset, performance evaluation metric, and experimental results analysis. Finally, the main conclusions are given in Section 5.

2. Related work

In this section, we provide a comprehensive introduction for previous works in the fields of information extraction based on span and contrastive learning in natural language processing.

2.1. Information extraction based on span

Span-based approaches take one or more words of a sentence sequence as span units, generate all possible spans, and model the sentence sequence at the span level. Getting rid of inherent sequential characteristic of sentence sequences, the span-based method can select overlapping entities and represent features, which can alleviate the error propagation and entity overlap problems caused by token-level sentence sequence modeling. Dixit et al. [18] proposed a span-based model for joint extraction of entities and relations using Bidirectional Long Short-term Memory (Bi-LSTM) network and achieved a better performance. Luan et al. [19] proposed a multi-task classification framework based on shared span representation, and constructed a scientific literature knowledge graph on the scientific literature abstract dataset SCIERC through tasks such as entity recognition, relationship classification and reference resolution. In order to enhance the interaction between different tasks, Luan et al. [20] proposed DYGIE, an information extraction framework based on dynamic span graph, which captured the interaction information between spans through graph propagation and improved the performance of the model without requiring additional syntactic analysis and processing. Based on the Bidirectional Encoder Representations from Transformers (BERT) model, Eberts et al. [21] realized the joint extraction of entities and relations through strong negative sample sampling, span filtering and local context representation. Ding et al. [22] applied the joint extraction method to a specific military field, proposed a hybrid model integrating span method and graph structure, and improved the extraction performance of the model by combining the vocabulary and syntactic knowledge of the specific field.

2.2. Contrastive learning in natural language processing

As a discriminative self-supervised learning, the contrastive learning is to leverage the similarity or dissimilarity of data samples to learn an encoder with supervised information. The encoder adopts the similar feature representation for the same type of data and different feature representations as far as possible for different types of data to obtain more effective feature representations for downstream tasks [23].

In terms of text presentation tasks, Giorgi et al. [24] applied contrastive learning to feature representation of sentence with unsupervised learning, and proved its feasibility in sentence representation task through experiments. In order to obtain a better sentence representation, Gao et al. [25] proposed a training method based on contrastive learning. This approach enhances sentence representation through specific data augmentation and contrastive loss function construction, and achieves a better performance in the downstream 7 Semantic Textual Similarity tasks (STS). To alleviate the collapse issue of BERT, Yan et al. [26] proposed a sentence representation transfer framework based on contrastive learning. By constructing comparative learning tasks, the framework fine-tunes BERT models on unlabeled datasets of the target domain, and generates sentence representations that are more suitable for downstream tasks.

In terms of entity and relation extraction tasks, Peng et al. [27] proposed an entity-masked comparative pre-training framework for relation extraction to capture sentence context information and entity type information. Firstly, the framework uses distant supervision method to generate positive and negative samples through external knowledge graphs. Then, sentences with the same relation are regarded as positive samples, and sentences with different relation are regarded as negative samples.

Finally, a randomly entity-masked method is used for pre-training to obtain a better relation representation. As the pre-trained model cannot capture factual knowledge in the text, Qin et al. [28] proposed a comparative learning framework in the pre-training phase to enhance understanding of entities and their semantic relations through Entity Discrimination and Relation Discrimination. Su et al. [29] improved the text representation of BERT model by means of comparative learning through data augmentation methods such as synonym replacement, random exchange and random deletion to enhance the effect of biomedical relation extraction.

3. Proposed method

3.1. Task modeling and model overview

The task of the span-level software knowledge triplet extraction proposed in this paper is to automatically identify all possible software knowledge entity spans from software knowledge community texts, predict their corresponding entity types, and classify the semantic relations of entity span pairs according to the predefined relation types, so as to obtain software knowledge triplets. In this way, the task can be formally defined as a 7-tuple $SKG=\left(X,S,Y_e,Y_r,\delta,\varepsilon,NA\right)$, where:

(1) $X=\left(x_1,x_2,\ldots,x_n\right)$ is an input sentence of the software knowledge community text;

(2) $S=\left(s_1,s_2,\cdots,s_n\right)$ is a candidate span set which generated by enumerating *X*;

(3) $Y_e(s_i)\in\delta\bigcup\{\text{NA}\}$ is a function to predict the entity type of candidate span instance s_i and generate a set $E=(e_1,e_2,\cdots,e_{|E|})$, $s_i=\left(x_i,x_{i+1},\ldots,x_{i+k}\right)$;

(4) $Y_r(e_i,e_j)\in\varepsilon\bigcup\{\text{NA}\}$ is a function to predict the semantic relation type of entity pairs (e_i,e_j) and generate a set $R=(r_1,r_2,\cdots,r_{|R|}),e_i,e_j\in E$;

(5) δ is a set of predefined entity types;

(6) ε is a set of predefined relation types;

(7)NA is a set of non-entity or no-semantic relation.

For example, given the sentence of the software knowledge community text "*GetHashCode is Method of Base Object Class of .NET Framework.*", the goal of software knowledge triplet extraction is to accurately identify entity pairs (e_i,e_j) : "*GetHashCode*" and "*.net Framework*", and predict the relation r_{ij} of entity pairs as "*inclusion*", thereby obtaining software knowledge relation triplets $<e_i,r_{ij},e_j>$.

According to the task definition mentioned above, we propose a novel hybrid model for software knowledge triplet extraction, named SCL-SKG, which based on span-level contrastive learning. The architecture of the SCL-SKG is shown in Figure 1.

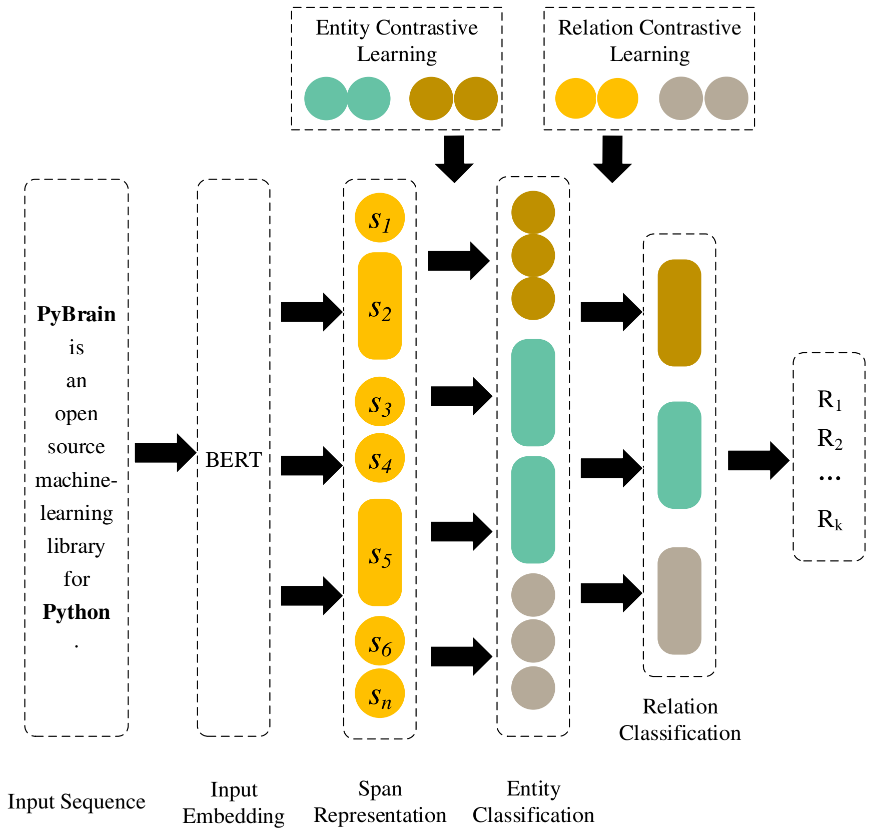

Figure 1: Overview of the proposed model.

3.2. BERT contextualized word embedding

Software knowledge community text is the user-generated content, which not only has social features such as repetitive content, loose structure and irregular spelling, but also has software domain features such as non-uniform naming, complicated terminology and weak semantic features. We use SWBERT [8], a pre-trained model in the field of software engineering, to encode the input sentence and capture the dynamic word embedding. The detailed description is as follows:

(1) For sentence sequence $X=(x_1,x_2,\ldots,x_n)$, the corresponding Token sequence is obtained by adding the identifiers [CLS] and [SEP] at the beginning and ending of the sentence sequence.

(2) For each token in the Token sequence, token embedding, segment embedding and position embedding are generated. After summing up these three embeddings, the input embedding of BERT is obtained: $E=(E_1,E_2,\cdots,E_n)$.

(3) After feature encoding, the dynamic word embedding of sentence sequence X is obtained:

$$H=[h_1,h_2,\cdots,h_n]=SWBERT(x_1,x_2,\cdots,x_n). \tag{1}$$

After the pre-trained model SWBERT encoding, a word embedding sequence with length n+1 is obtained: $W = (w_{cls}, w_1, w_2, ..., w_n)$, where w_{cls} represents the classification information of the sentence sequence.

3.3. Token and span representation

In view of the entity overlap problem in the software knowledge community text, inspired by relevant work [18][21], we constructed a span-level sentence sequence representation layer in the SCL-SKG and modeled sentence sequences as span units. Generally, span-based methods generate span representations by iterating all the words in the sentence sequence, which incurs model computational overhead. Therefore, we first filter sentence sequences, removing words with less meaning and reserving words such as verbs and nouns. Then, all words of sentence sequences are iterated to produce span representations with different lengths, resulting in a set of entity span: $S=(s_1,s_2,\cdots,s_n)$. The entity span instance is represented as $s_i=(x_i,x_{i+1},...,x_{i+k})$, where k is the length of the span, indicating the number of words contained in the entity span.

For example, for the sentence sequence of software knowledge community text: "*GetHashCode is Method of Base Object Class of .NET Framework.*", the sentence sequence: "*GetHashCode Method Base Object Class . NET Framework.*" is obtained after filtering, and the generated entity span sets are: "*GetHashCode*", "*Method*", "*GetHashCode Method*", "*Method Base*", "*Object*", "*Base Object*", "*Class*", "*.NET*", "*.NET Framework*", etc.

Therefore, the SCL-SKG model generates the entity span of sentence sequences through the entity span representation layer, and generates abundant positive and negative samples of entity span, which provides a data augmentation method for the next step of entity contrastive learning.

3.4. Entity classification based on contrastive learning

In order to obtain distinctive feature representation of entity span and improve the accuracy of entity span classification prediction, we propose an entity contrastive learning at entity classification layer. Therefore, the entity classification layer of the SCL-SKG model includes two steps: entity contrastive learning and entity classification.

3.4.1. Supervised span-level entity contrastive learning

In order to make use of the label information of software knowledge entities, we extend the contrastive self-supervised learning method and propose a supervised span-level entity contrastive learning to obtain the entity span feature representation that is more suitable for downstream tasks. Different from the self-supervised contrastive learning, the supervised span-level entity contrastive learning combines entity label information and data augmentation methods to generate multiple positive and negative views of the original data samples, and uses the contrastive loss function constraint model to learn the feature representation of entity span that is more suitable for classification tasks.

The following is a detailed description of the components involved in the supervised span-level entity contrastive learning.

(1) Data augmentation: Compared with the image processing field, the data augmentation in the natural language processing field is more difficult, as it includes random deletion of words, random insertion of words, random exchange of words and synonym-antonym replacement, etc. [30]. However, these methods are likely to interfere with the structure and semantic information of sentences, and if they are directly applied to the downstream tasks such as entity extraction and relation extraction, it will affect the performance of the model.

Based on the entity span set in the span representation layer, we use the labels of software knowledge entities for data augmentation to generate multiple positive and negative samples of entity span instances. Specifically, supervised span-level entity contrastive learning regards entity spans with the same type as positive samples and constructs positive sample set $P(i)$, and regards other types of entity span or non-entity span in the same batch as negative samples and constructs negative sample set $N(i)$.

(2) Encoder: In the encoder component, the SCL-SKG model uses the pre-trained model SWBERT to transform the input sentence sequence $X=(x_1,x_2,\dots,x_n)$ into dynamic word embedding, and extract text features, which can be expressed as:

$$h_i = f(x_i)=\text{SWBERT}(x_i)\,. \tag{2}$$

(3) Projection network: Referring to Chen's work in the image field [31], the SCL-SKG model utilizes Multi-Layer Perceptron (MLP) to project the embedding to another representation space in the projection network component, so as to obtain better feature representation in the training phase, which can be expressed as:

$$z_i = g(h_i) = W^2\sigma(W^1 h_i)\,. \tag{3}$$

Where, W^1 *and* W^2 represent the weights of hidden layers, σ is *ReLU* activation function.

(4) Contrastive loss function: Since self-supervised contrastive learning cannot deal with type information of entities and will lead to multiple positive samples problem, we refer to the work of Khosla[32] and extend the self-supervised contrastive loss function to obtain the loss function of supervised span-level entity contrastive learning, which is expressed as:

$$L^{ec} = \sum_{i \in B(i)} \frac{-1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp(z_i \cdot z_p / \tau)}{\sum_{n \in N(i)} \exp(z_i \cdot z_n / \tau)} . \quad (4)$$

Where, z_i is the span instance of the current entity, z_p is the positive sample instance of z_i, z_n is the negative sample instance of z_i, $B(i)$ is the sample set in batch, $|P(i)|$ is cardinality of positive sample set, $N(i)$ is the set of negative samples, τ temperature parameter, the symbol • denotes the inner product operation of similarity calculation.

Thus, the span-level entity contrastive learning can be described as Algorithm 1.

Algorithm 1 Span-level entity contrastive learning algorithm

```
Input: Entity span instance $s_i$, pre- trained model SWBERT, Projection network $g$
Output: The embedding $h_i$ of entity span instance $s_i$
1. Begin
2.     Get the embedding $h_i$ of the entity span instance $s_i$        // Formula (2)
3.     Project $h_i$ into another representation space, and get $z_i$        // Formula (3)
4.     Use the data augmentation to generate positive and negative sample of $z_i$ : $z_p$, $z_n$
5.     Calculate the entity contrast loss $L^{ec}$        // Formula (4)
6.     Update the parameters of SWBERT and network $g$ to minimize $L^{ec}$
7.     Return the embedding $h_i$ of entity span instance $s_i$
8. End
```

3.4.2. Entity classification

The goal of entity classification is to predict the type of candidate entity span and filter non-entity span at the same time. Therefore, the entity classification consists of the following two steps:

(1) Embedding concatenation. After span-level entity contrastive learning, the final embedding of candidate entity span s_i is obtained, which can be expressed as:

$$s_i = [h_i; s_{width}; w_{cls}] . \quad (5)$$

Where, h_i is the embedding of entity span instance, s_{width} is the embedding of length of entity span instance, and w_{cls} is the special classification information.

(2) Entity type prediction. After embedding concatenation, the candidate entity span s_i is fed into a Softmax layer for entity type prediction:

$$e_i = \text{softmax}(W_i \cdot s_i + b_i) . \quad (6)$$

Where, W_i represents the weight matrix, and b_i represents the bias units.

3.5. Relation classification based on contrastive learning

In order to obtain a distinctive feature representation of candidate entity pairs that is more suitable for relation classification, we propose a supervised span-level relation contrastive learning at relation classification layer. Similar to the above entity classification, the relation classification layer of the SCL-SKG model includes two steps: relation contrastive learning and relation classification.

3.5.1. Supervised span-level relation contrastive learning

Compared with the supervised entity contrastive learning, the Encoder and Projection network of the supervised relation contrastive learning remain unchanged, and the data augmentation and contrastive loss function are different.

In the data enhancement component, supervised relation contrastive learning regards entity pairs with the same relation type as positive samples and constructs positive sample set $P(i)$, and regards other relation types of entity pairs or non-relation in the same batch as negative samples and constructs negative sample set $N(i)$.

Therefore, the contrastive loss function of supervised relation contrastive learning is defined as follows:

$$L^{rc} = \sum_{i \in B(i)} \frac{-1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp(z_i \cdot z_p / \tau)}{\sum_{n \in N(i)} \exp(z_i \cdot z_n / \tau)}. \tag{7}$$

Where, z_i is the instance of the current entity pairs, z_p is the positive sample instance of z_i, z_n is the negative sample instance of z_i, $B(i)$ is the set of candidate entity pair in batch, $|P(i)|$ is cardinality of positive sample set, $N(i)$ is the set of negative samples, τ temperature parameter, and the symbol • denotes the inner product operation of similarity calculation.

Similarly, span-level relation contrastive learning can be described as Algorithm 2.

Algorithm 2 Span-level relation contrastive learning algorithm

Input: Candidate entity pair (s_i, s_j), pre- trained model SWBERT, Projection network g
Output: The embedding h_i of candidate entity pair (s_i, s_j)

```
1. Begin
2.     Get the embedding $h_i$ of candidate entity pair $(s_i, s_j)$
3.     Project $h_i$ into another representation space, and get $z_i$
4.     Use the data augmentation to generate positive and negative sample of $z_i$ : $z_p$, $z_n$
5.     Calculate the entity contrast loss $L^{rc}$          Formula (7)
6.     Update the parameters of SWBERT and network $g$ to minimize $L^{rc}$
7.     Return the embedding $h_i$ of candidate entity pair
8. End
```

3.5.2. Relation classification

The goal of relation classification is to predict the type of candidate entity pair and filter non-relation at the same time. Therefore, the relation classification consists of the following two steps:

(1) Embedding concatenation. After span-level relation contrastive learning, the final embedding of entity pair s_{ij} is obtained, which can be expressed as:

$$s_{ij} = [h_i ; c_{i,j}]. \tag{8}$$

Where, h_i is the embedding of entity pair, and c_{ij} is the context of span pair.

(2) Relation type prediction. After embedding concatenation, the candidate entity pair s_{ij} is fed into a fully connected layer for relation classification:

$$r_{ij} = \sigma(W_{ij} \cdot s_{ij} + b_{ij}) . \tag{9}$$

Where, σ is activation function, W_{ij} represents the weight matrix, and b_{ij} represents the bias units.

In summary, the software knowledge triplet extraction with span-level contrastive learning is described as Algorithm 3.

Algorithm 3 Software knowledge triplet extraction with span-level contrastive learning

```
Input: Sentence sequence of software knowledge community text $X=(x_1,x_2,...,x_n)$
Output: Software knowledge entity relation triplet
1.  Begin
2.      for each epoch do
3.          for each batch do
4.              The word embedding of sentence sequence $X$ is generated by SWBERT
5.              Generates a span representation of the sentence sequence $S=(s_1,s_2,...,s_n)$
6.                  for $s_i$ from $S$ do
7.                      Call Algorithm 1 to get the embedding $h_i$ of entity span instance $s_i$
8.                      The embedding of $s_i$ is obtained by embedding concatenation      //Formula (5)
9.                      Predict the type of entity span instance $s_i$      //Formula (6)
10.                 endfor
11.             The software knowledge entity set $E$ is obtained, and the entity pair instance is selected $(s_i, s_j)$
12.                 for $s_i$, $s_j$ from $S$ do
13.                     Call Algorithm 2 to get the embedding $h_i$ of entity pair instance $(s_i, s_j)$
14.                     The embedding of $(s_i, s_j)$ is obtained by embedding      //Formula (8)
15.                     Predict the type of relation for entity pair instance $(s_i, s_j)$      //Formula (9)
16.                 endfor
17.             Return the software knowledge entity relation triplet
18.         endfor
19.     endfor
20. End
```

As shown in Algorithm 3, firstly, a pre-trained language model SWBERT in software engineering was used as the input feature encoder to obtain the dynamic word embedding of the sentence sequence. Then, the sentence sequence is modeled with spans as the unit to generate rich entity span representation to avoid the problem that overlapping entities cannot be selected. Finally, the supervised contrastive learning is introduced into the entity classification and relation classification tasks, and the entity span and entity pair feature representation are obtained by using data augmentation and contrastive loss function, so as to improve the performance of entity classification and relation classification.

4. Experiments

In order to evaluate the performance of SCL-SKG model proposed in this paper, the ablation experiments and comparative experiments with the benchmark models in the field of joint entity and relation extraction were carried out. SCL-SKG model was implemented in Python using the deep learning framework PyTorch. It was specifically configured as Intel Xeon Gold 5117 processor, 2.0 GHz clock speed, NVIDIA Tesla T4 GPU, 16GiB display memory, and all the experiments in this paper were conducted in this experimental environment.

4.1. Dataset

Due to the lack of available annotated dataset for the software knowledge triplet extraction task, we build an annotated dataset based on the text of StackOverflow with reference to related research works[3][8]. For the types of software knowledge entity and relation, we constructed 8 predefined entity types and 8 predefined relation types. The detailed information is shown in Table 1.

Table 1: The types of software knowledge entity and relation.

	Name	Meaning	Instance	Abbreviation
Entity types	Programming Language		Java, Python,C	prla
	Software Platform		Weblogic,Dolphin	plat
	Software API		QueryManager, isNumeric	api
	Software Tool		Pycharm,Firebug	tool
	Software Library		jQuery,NumPy	lib
	Software Framework		Hibernate,Django	fram
	Software Standard		HTTP, utf-8	stan
	Development Process		Umlet,LoadRunner	depr
Relation types	Use	relation of using	Windows 10,Cortana	use
	Inclusion	relation of including	Python,PyBrain	inc
	Brother	relation of brother	C,Java	bro
	Consensus	relation of synonym	JavaScript,JS	con
	Semantic	other semantic relation	Virtual Network,VPN	sem

In terms of data set annotation, we use the JavaScript Object Notation (JSON) file format to annotate information such as sentence sequence, entity type, start and end positions of entity span, relation type, header entity and tail entity to form software knowledge triplet annotated data sets. In order to ensure the scientific and reasonable results of model experiment, the dataset is divided into training set, verification set and test set according to the ratio of 7:1:2 for the experiment of software knowledge triplet extraction. The detailed information of the dataset is shown in Table 2.

Table 2: The detail of the dataset.

	Training set	Verification set	Test set	Total
Sentence	15016	196	3801	19013
Entity	34592	429	8748	43769
Relation	19875	234	5074	25183

4.2. Parameter settings

For the span-based module in SCL-SKG model, the word embedding dimension of pre-trained language model SWBERT is set as 768 dimensions, Batch size is set to 3, the maximum of span is set to 10, and the maximum value of entity span negative sample and relation negative sample is set to 100, Adam is used as the optimizer, the initial learning rate is set at 5e-5. For the contrastive learning module in SCL-SKG model, contrastive loss is used as the loss function of the model, and the temperature parameter is set to 0.1. The setting of relevant hyper-parameters is shown in Table 3.

Table 3: Hyper-parameters of the proposed model.

Modules	Names	Value
Span-based module	Word embedding dimension	768
	Batch size	3
	Learningrate	5e-5
	Max_span_size	10
	Num_epoch	100
	Num_negative_entiy	100
	Num_negative_relation	100
	Optimizer	Adam
	Dropout	0.5
Contrastive Learning module	τ	0.1

4.3. Evaluation metrics

The software knowledge triplet extraction model SCL-SKG involves two subtasks: entity recognition and relation extraction. The evaluation metrics of the extraction results are as follows: If both the boundary and the type of software knowledge entity span are predicted correctly, the result of entity recognition is correct; if the boundary, type and semantic relation of software knowledge entities are predicted correctly, the result of relation extraction is correct.

General evaluation metrics in information extraction task are selected to evaluate the performance of the model, including precision rate (P), recall rate (R) and F1 score (F1). Precision rate (P) represents the percentage of correctly recognized samples in all recognized samples in the model recognition results; the recall rate (R) represents the percentage of correctly recognized samples in the number of all correct samples; F1 score is the weighted harmonic average of precision rate (P) and recall rate (R), which is used as the comprehensive performance evaluation index of the model. The formal expression of each evaluation index is as equations (10)-(12):

$$P = \frac{T_P}{T_P + F_P}, \tag{10}$$

$$R = \frac{T_P}{T_P + F_n}, \tag{11}$$

$$F1 = \frac{2 \times P \times R}{P + R}. \tag{12}$$

Where T_P (True Positive) represents the number of correct relation types that are recognized as positive examples by model, F_P (False Positive) represents the number of the wrong relation types that are recognized as positive examples by model and F_N (False Negative) represents the correct number of relation types that are recognized as the negative examples by model.

4.4. Results and discussions

To evaluate the performance of SCL-SKG model proposed in this paper, three state-of-the-art joint extraction models were selected for comparative experiments from two aspects: parameter sharing-based approach and joint decoding-based approach.

Multi-head model [33] is a joint entity and relation extraction model based on shared parameter approach. This model uses BILOU annotation method and CRF decoding to realize entity extraction, use multi-head selection algorithm and sigmoid layer to realize relation extraction.

SPERT model [21] is a joint entity and relation extraction model based on shared parameter approach. This model abandons the traditional method based on BIO/BILOU annotation, and uses the pre-trained language model BERT to obtain the word embedding of sentence sequence, and implements the joint entity and relation extraction by enumerating all possible entity spans in sentence sequence.

NovelTagging model [34] is a joint entity and relation extraction model based on joint decoding approach. This model implements an end-to-end joint entity and relation extraction based on a new sequence annotation framework and LSTM network.

The experimental results are shown in Table 4.

Table 4: Experimental results on software knowledge dataset

Model	Entity			Relation		
	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)
Multi-head[33]	68.20	65.23	66.68	61.03	56.82	58.85
NovelTagging[34]	75.67	65.39	70.16	67.15	63.31	65.17
SPERT[21]	83.71	67.17	74.53	72.27	69.64	70.93
SCL-SKG	82.19	78.91	**80.52**	80.37	76.03	**78.14**

From the experiment results, it can be seen that the F1 score of the SCL-SKG model is higher than the other three baseline models, and has achieved a better performance. From the perspective of software knowledge entity extraction task, compared with Multi-head model, precision rate and F1 score are improved by 13.99% and 13.84%, respectively. Compared with NovelTagging model, precision rate and F1 score are improved by 6.52% and 10.36%, respectively. Compared with SPERT model, precision rate decreased by 1.52%, and F1 score increased by 5.99%.

From the perspective of software knowledge entity relation extraction task, compared with Multi-head model, precision rate and F1 score are improved by 19.34% and 19.29%, respectively. Compared with NovelTagging model, precision rate and F1 score are improved by 13.22% and 12.97%, respectively. Compared with SPERT model, precision rate and F1 score are improved by 8.1% and 7.21%, respectively.

Compared with the token-level approach, the span-based approach takes spans as the unit to model the sentence sequence, which can alleviate the entity overlap problem and improve the performance of the model. Therefore, SPERT model and SCL-SKG model which are span-based approach obtained the highest precision rate of entity extraction and relation extraction, respectively. Meanwhile, compared with the SPERT model, the SCL-SKG model introduced the contrastive learning based on the span approach, and the F1 score of entity extraction and relation extraction were increased by 5.99% and 7.21%, respectively.

In addition, the results of the SCL-SKG model for each predefined software knowledge entity type and relation type are shown in Figure 2 and Figure 3.

It can be seen from the results that the entity extraction task of SCL-SKG model has higher performance than the relation extraction task. In the task of entity extraction, the SCL-SKG model has better performance on entity types such as Software Tool, Software Library and Programming Language. The highest F1 score is obtained in the entity type of Software Platform, but the lowest F1 score appears in the entity type of Software Development Process. In the task of relation extraction, the highest F1 score was obtained in the relation type of Inclusion, while the lowest F1 score was obtained in the relation type of Semantic.

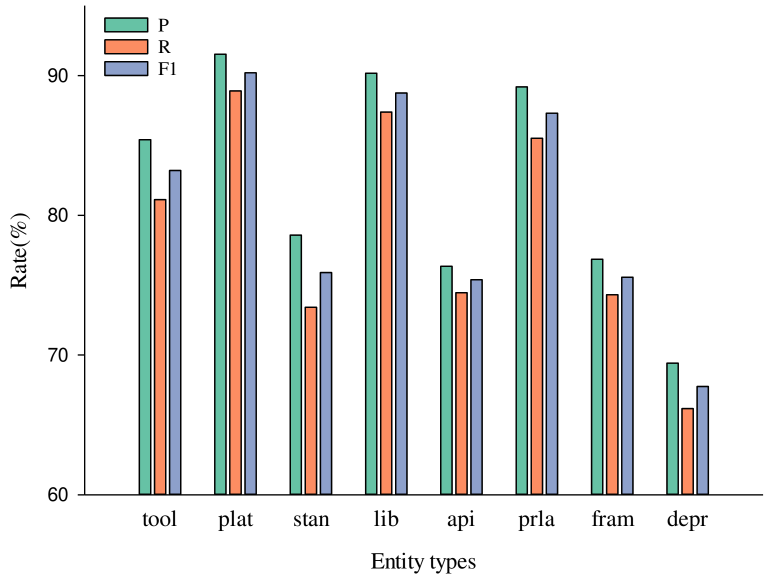

Figure 2: Extraction results for each entity type

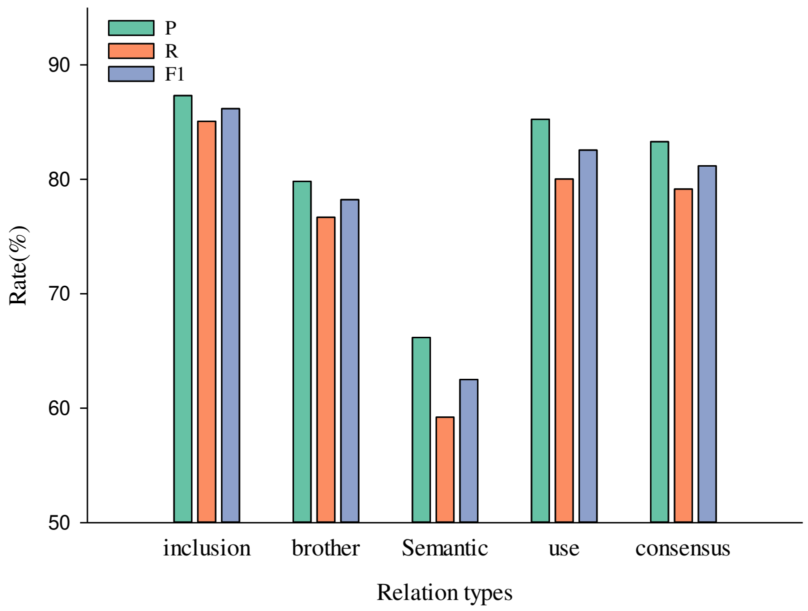

Figure 3: Extraction results for each relation type

4.5. Ablation experiment and analysis

The main goal of ablation experiments on SCL-SKG model is to verify the contribution of the proposed entity contrastive learning, relation contrastive learning, and pre-trained model SWBERT to software knowledge triplet extraction.

4.5.1. Contribution of contrastive learning to performance

We selected the SCL-SKG model as the benchmark model to evaluate the contribution of entity contrastive learning and relation contrastive learning to the software knowledge triplet extraction. The experimental results are shown in Table 5.

In Table 5, the SCL-SKG-NN model indicates that entity contrastive learning and relation contrastive learning are not introduced. The SCL-SKG-EC model indicates that only entity contrastive learning is introduced, the SCL-SKG-RC model indicates that only relation contrastive learning is introduced, the SCL-SKG-ALL model indicates that both entity contrastive learning and relation contrastive learning are introduced.

Table 5: Contribution of contrastive learning to model performance

Model	Entity contrastive learning	Relation contrastive learning	Entity			Relation		
			P%	R%	F1%	P%	R%	F1%
SCL-SKG-NN	×	×	68.43	59.61	63.72	65.32	58.34	61.63
SCL-SKG-EC	✓	×	81.97	76.69	79.24	69.74	62.57	65.96
SCL-SKG-RC	×	✓	69.11	60.93	64.76	71.51	65.72	68.49
SCL-SKG-ALL	✓	✓	82.19	78.91	**80.52**	80.37	76.03	**78.14**

According to the experiment results, after only introducing entity contrastive learning, the F1 score of entity extraction and relation extraction is increased by 15.52% and 4.33%, respectively. After only introducing relation contrastive learning, the F1 score of entity extraction and relation extraction are increased by 0.04% and 6.86%, respectively. After introducing both entity contrastive learning and relation contrastive learning, the F1 score of entity extraction and relation extraction reaches the highest.

Experimental results show that entity contrastive learning and relation contrastive learning can obtain feature representations of entity span and entity pair that are more suitable for downstream classification tasks, which is helpful to improve the performance of software knowledge triplet extraction.

4.5.2 Contribution of pre-trained model to performance

In order to evaluate the contribution of pre-trained language model SWBERT to software knowledge triplet extraction, SCL-SKG model is selected as the benchmark model, and the experimental results are shown in Table 6.

In Table 6, the model is labelled by symbol "✓", if the corresponding features representation is used; Otherwise, it is labelled "×". Among them, the model SCL-SKG-NN indicates that no pre-trained model is introduced, the model SCL-SKG-BERT indicates that the general domain BERT model is introduced, and the model SCL-SKG-SWBERT indicates that the pre-trained model for software domain SWBERT is introduced.

Table 6: Contribution of pre-trained model to model performance

Model	Pre-trained model	Entity			Relation		
		P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)
SCL-SKG-NN	×	77.26	69.71	73.29	71.84	68.27	70.01
SCL-SKG-BERT	✓	80.95	73.19	76.87	76.36	70.17	73.13
SCL-SKG-SWBERT	✓	**82.19**	78.91	**80.52**	80.37	76.03	**78.14**

According to the experimental results, the F1 score of entity extraction and relation extraction are increased by 3.58% and 3.12% respectively after the general domain BERT is introduced. The F1 score of entity extraction and relation extraction are increased by 7.23% and 8.13%, after the pre-trained model SWBERT is introduced. In addition, compared with the introduction of BERT, the F1 score of entity extraction and relation extraction increased by 3.65% and 5.01% respectively after the introduction of SWBERT.

4.6. Comparison with state-of-the-art models on public dataset

To further evaluate the performance of SCL-SKG model proposed in this paper, we compare SCL-SKG with state-of-the-art joint extraction models on three public datasets. The CoNLL04 dataset

[35] is an annotated data set for news articles, including 4 entity types and 5 relation types. The SciERC dataset [36] is derived from 500 abstracts of AI conference/workshop proceedings in four AI communities. The ADE dataset [37] is derived from medical reports from drug use, including 2 entity types and 1 relation type.

Following the evaluation method of previous work, we measure the macro-averaged values for the CoNLL04 dataset and the ADE dataset; and measure the micro-averaged values for SciERC dataset. The experimental results are shown in Table 7.

Table 7: Experimental results on public dataset

Dataset	Model	Entity			Relation		
		P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)
CoNLL04	Multi-head[33]	83.75	84.06	83.90	63.75	60.43	62.04
	Multi-head+AT[38]	-	-	83.61	-	-	61.95
	Table-filling[39]	81.20	80.20	80.70	76.00	50.90	61.00
	SPERT[21]	85.78	86.84	86.25	74.75	71.52	72.87
	SCL-SKG	86.45	87.35	**86.93**	76.13	71.31	**73.12**
SciERC	SciIE[40]	67.20	61.50	64.20	47.60	33.50	39.30
	DyGIE[20]	-	-	65.20	-	-	41.60
	DyGIE++[41]	-	-	67.50	-	-	48.40
	SPERT[21]	68.53	66.73	67.62	49.79	43.53	46.44
	SCL-SKG	68.24	66.28	67.21	50.37	44.69	**47.56**
ADE	BiLSTM+SDP[42]	82.70	86.70	84.60	67.50	75.80	71.40
	Multi-head[33]	84.72	88.16	86.40	72.10	77.24	74.58
	Multi-head+AT[38]	-	-	86.73	-	-	75.52
	SPERT[21]	88.99	89.59	89.28	77.77	79.96	78.84
	SCL-SKG	87.53	90.25	88.89	75.35	80.33	77.91

According to the experimental results, SCL-SKG model achieve performance improvement of entity extraction and relation extraction on the CoNLL04 dataset, the F1 score of entity extraction and relation extraction are increased by 0.7% and 0.3 %, respectively. For SciERC dataset, the relation extraction performance also achieves improvement, the F1 score of relation extraction is increased by 1.1%. Compared with SPERT model, SCL-SKG model does not achieve performance improvement of entity extraction and relation extraction on the ADE dataset.

4.7. Analysis of joint training methods

The loss function of the proposed SCL-SKG is composed of four parts: entity contrastive learning loss L^{ec}, entity classification loss L^{e}, relation contrastive loss L^{rc} and relation classification loss L^{r}. Among them, entity classification loss L^{e} adopts Categorical Cross Entropy as loss function, and the relation classification loss L^{r} adopts the Binary Cross Entropy as the loss function.

In order to obtain the best joint training result of the proposed SCL-SKG, we tried three different joint training methods, namely adding loss function, multiplying loss function and linear combination of loss function. The specific formula is as follows:

$$L = L^{e} + L^{ec} + L^{r} + L^{rc}, \tag{13}$$

$$L = L^{e} * L^{ec} + L^{r} * L^{rc}, \tag{14}$$

$$L = L^{ec} + \lambda(\cdot)L^{e} + L^{rc} + \lambda(\cdot)L^{r}. \tag{15}$$

Where, the adding loss function represents the sum of four losses, such as entity contrastive loss L^{ec}, entity classification loss L^{e}, relation contrastive loss L^{rc} and relation classification loss L^{r}, as shown in Formula 13. The multiplying loss function means entity contrastive loss multiplied by entity classification loss, and relation contrastive loss multiplied by relation classification loss, as shown in Formula 14. Linear combination of loss function means that a linear function is added to the entity classification loss and relation classification loss, as shown in Formula 15.

As can be seen from Figure 4 and Figure 5, the method of multiplication of loss functions will make the model fail to converge and achieve poor results. The method of adding loss function and linear combination of loss function can complete the model training and testing, and the linear combination method achieves better results.

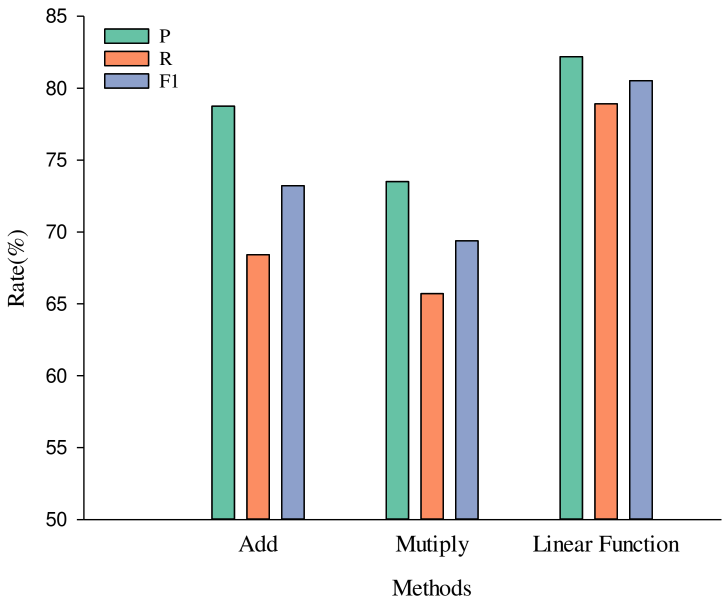

Figure 4: Joint training methods for entity extraction

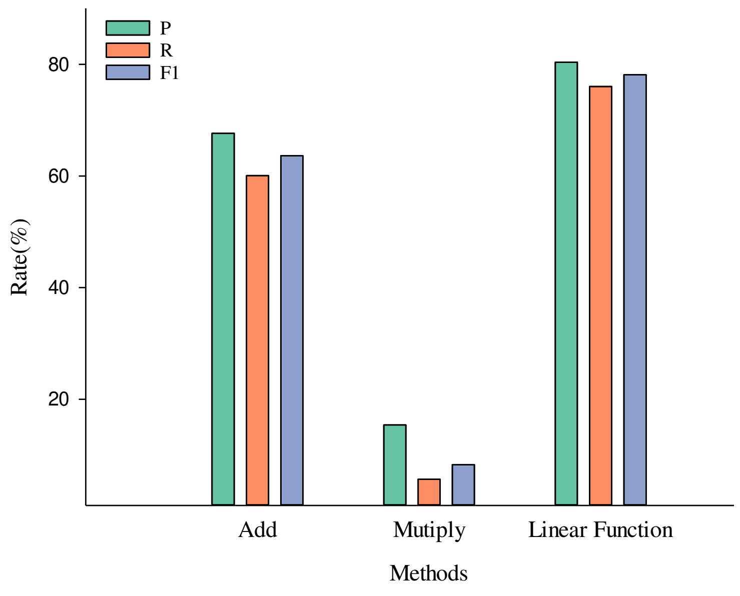

Figure 5: Joint training methods for relation extraction

4.8. Case analysis

The above experimental results show that the software knowledge triplet extraction model SCL-SKG based on span-level contrastive learning achieve better performance, and construct a software knowledge graph which contain 43769 entity instances and 25183 relation instances. The overview diagrams of the software knowledge graph with 50 nodes and 1000 nodes are shown in Figure 6 and Figure 7.

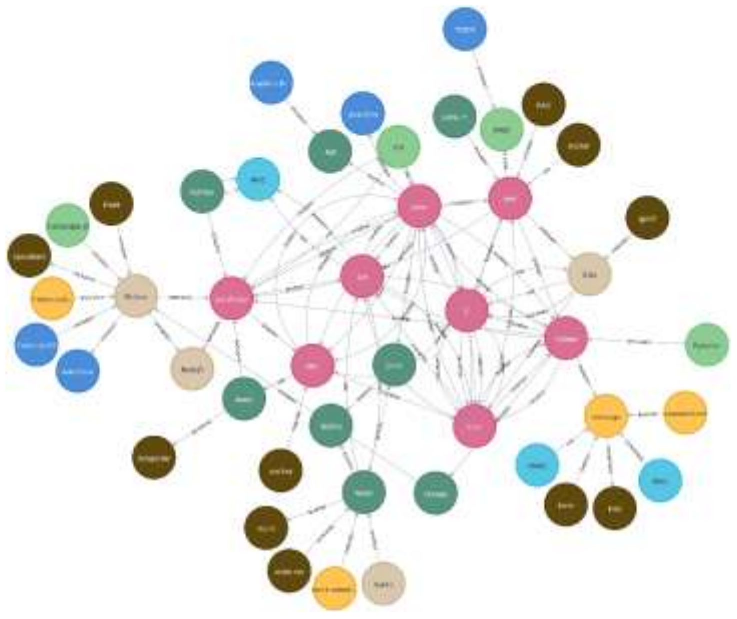
Figure 6: The software knowledge graph with 50 nodes

Figure 7: The software knowledge graph with 1000 nodes

Although the proposed SCL-SKG has achieved good results, there are still some specific problems. The specific case of the SCL-SKG is analyzed below, and the analysis results are shown in Table 8, where the symbol "[]" represents the extracted software knowledge entity.

Table 8: Case analysis of software knowledge triplet extraction

	Results
Case 1	SpriteKit is **[[Apples] framework]** for creating 2D games for **[[IOS] 7]**, **[[macOS] 10.9]**, **[[tvOS] 9]** and **[[watchOS] 3]**.
Case 2	StarlingFramework is an **[ActionScript 3]** library for hardware accelerated 2D graphics.
Case 3	**[BlackBerry]** offers a variety of development tools, including the **[BlackBerry Dynamics SDK]**, **[Cylance REST APIs]**,**[BlackBerry Workspaces APIs]** and SDKs, BlackBerry QNX development and BlackBerry UEM REST APIs.
Case 4	**[CUDA]** (**[Compute Unified Device Architecture]**) is a parallel computing platform and programming model for NVIDIA GPUs (Graphics Processing Units).

In the Case 1, SCL-SKG model can not only extract software knowledge entities "Apples" and "IOS", but also accurately extract software knowledge entities "Apples Framework" and "IOS 7", indicating that the SCL-SKG model can effectively solve the problem of entity overlap.

In the Case 2, because the boundary of entity span is wrong, SCL-SKG model does not accurately identify the software knowledge entity "ActionScript 3 Library", which causes the relation extraction error.

In the Case 3, SCL-SKG model extracted software knowledge entities "BlackBerry", "BlackBerry

Dynamics SDK" and "Cylance REST APIs", and accurately extracted the relation of "BlackBerry" and "BlackBerry Dynamics SDK" as "Inclusion". Meanwhile, although the relation of "BlackBerry Dynamics SDK" and "Cylance REST APIs" is not labeled in the training dataset, the model predicts that the relation of the two entities is "Brother".

In the case 4, SCL-SKG model identifies the Software knowledge entities "CUDA" and "Compute Unified Device Architecture" as the types of "Software Platform" and "Software Tool" respectively, resulting in entity extraction errors and the "Consensus" relation is not correctly identified.

5. Conclusion

In view of the problems of task dependence in traditional Pipeline method and entity overlap in software knowledge community text, we proposes a software knowledge triplet extraction method based on span-level contrastive learning, and takes software knowledge community StackOverflow as an example to carry out experiments and analysis. The experimental results show that the span-level contrastive learning method can alleviate the overlap problem of software knowledge entities by modeling sentence sequences with spans as the unit. At the same time, supervised entity contrastive learning and relation contrastive learning can obtain the enhanced feature representation of entity span and entity pair, which is helpful to improve the performance of software knowledge entity and relation classification.

Acknowledgements

We thank the anonymous reviewers and the editors for their suggestions. This work is supported by the National Natural Science Foundation of China (Grant no. 62266054), Science and Technology Program of Yunnan Province (Grant no. 202101AT070095).

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Mingjing Tang: Writing-original draft, Data curation, Visualization. **Shu Zhang:** Writing-review & editing. **Ming Zheng:** Conceptualization, Methodology. **Zifei Ma:** Software, Validation. **Wei Gao:** Investigation, Supervision.

Data availability

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

ORCID

Mingjing Tang https://orcid.org/0000-0003-0435-8854

References

[1] Yin G, Wang T, Liu BX, et al. Survey of Software Data Mining for Open Source Ecosystem[J]. Journal of Software, 2018, 29(8): 2258-2271.

[2] Tabassum J, Maddela M, Xu W, Ritter A. Code and Named Entity Recognition in StackOverflow[C]. Proc. 58th Annual Meeting of the Association for Computational Linguistics (ACL), Online, 2020: 4913-4926.

[3] Ye DH, Xing ZC, Foo CY, et al. Software-Specific Named Entity Recognition in Software Engineering Social Content[C]. Proc. 23th International Conference on Software Analysis, Evolution, and Reengineering (SNER), Osaka, Japan, 2016: 90-101.

[4] Reddy MVPR, Prasad PVRD, Chikkamath M, et al. NERSE: named entity recognition in software engineering as a service[C]. Proc. Australian Symposium on Service Research and Innovation, 2019: 65-80.

[5] Lv WQ, Liao ZF, Liu SZ, et al. MEIM: a multi-source software knowledge entity extraction integration model[J]. Computers, Materials & Continua, 2021, 66(1): 1027-1042.

[6] Zhu JG, Shen BJ, Cai XY, et al. Building a large-scale software programming taxonomy from stackoverflow[C]. Proc. International Conference on Software Engineering and Knowledge Engineering (SEKE), Pittsburgh, PA, USA, 2015: 391-396.

[7] Zhao XJ, Xing ZC, Kabir MA, et al. HDSKG: Harvesting Domain Specific Knowledge Graph from Content of Webpages[C]. Proc. 24th International Conference on Software Analysis, Evolution and Reengineering (SANER), Klagenfurt, Austria, 2017: 56-67.

[8] Tang MJ, Li T, Gao W, Xia Y. AttenSy-SNER: software knowledge entity extraction with syntactic features and semantic augmentation information [J]. Complex & Intelligent Systems, 2022: 1-15.

[9] Geng ZQ, Zhang YH, Han YM. Joint entity and relation extraction model based on rich semantics [J]. Neurocomputing, 2021, 429:132-140.

[10]Han X, Gao TY, Lin YK,et al. More Data, More Relations, More Context and More Openness: A Review and Outlook for Relation Extraction[C]. Proc. 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing, Suzhou, China, 2020: 745-758.

[11]Ye HB, Zhang NY, Deng SM, et al. Contrastive Triple Extraction with Generative Transformer[C]. Proc. 35th AAAI Conference on Artificial Intelligence (AAAI), Online, 2021:14257-14265.

[12]Miwa M, Bansal M. End-to-end relation extraction using LSTMs on sequences and tree structures[C]. Proc. Meeting of the Association for Computational Linguistics (ACL), 2016: 1105-1116.

[13]Zheng SC, Hao YX, Lu DY, et al. Joint entity and relation extraction based on a hybrid neural network[J]. Neurocomputing, 2017, 257: 59-66.

[14]Li F, Zhang MS, Fu GH, et al. A neural joint model for entity and relation extraction from biomedical text[J]. BMC Bioinformatics, 2017, 18(1): 198.

[15]Zheng SC, Wang F, Bao HY,et al. Joint Extraction of Entities and Relations Based on a Novel Tagging Scheme[C]. Proc. 55th Annual Meeting of the Association for Computational Linguistics (ACL), Vancouver, Canada, 2017: 1227-1236.

[16]Bekoulis G, Deleu J, Demeester T, et al. Joint entity recognition and relation extraction as a multi-head selection problem [J]. Expert Systems with Applications, 2018, 114: 34-45.

[17]Zeng XR, Zeng DJ, He SZ, et al. Extracting Relational Facts by an End-to-End Neural Model with Copy Mechanism[C]. Proc. 56th Annual Meeting of the Association for Computational Linguistics (ACL), Melbourne, Australia, 2018: 506-514.

[18]Dixit K, Al-Onaizan Y. Span-Level Model for Relation Extraction[C]. Proc. 57th Annual Meeting of the Association for Computational Linguistics (ACL), Florence, Italy, 2019: 5308-5314.

[19]Luan Y, He LH, Ostendorf M, et al. Multi-Task Identification of Entities, Relations, and Coreference for Scientific Knowledge Graph Construction[C]. Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP), Brussels, Belgium, 2018: 3219-3232.

[20]Luan Y, Wadden D, He LH,et al. A general framework for information extraction using dynamic span graphs[C]. Proc. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, Minnesota, 2019: 3036-3046.

[21]Eberts M, Ulges A. Span-based Joint Entity and Relation Extraction with Transformer Pre-training[C]. Proc. 24th European Conference on Artificial Intelligence, Santiago de Compostela, Spain, 2019: 2006-2013.

[22]Ding K, Liu SS, Zhang YH, et al. A Knowledge-Enriched and Span-Based Network for Joint Entity and Relation Extraction [J]. Computers, Materials & Continua, 2021, 68(1): 377-389.

[23]Liu X, Zhang FJ, Hou ZY, et al. Self-supervised Learning: Generative or Contrastive [J]. IEEE Transactions on Knowledge and Data Engineering, 2021, doi: 10.1109/TKDE.2021.3090866.

[24]Giorgi J, Nitski O, Wang B, et al. DeCLUTR: Deep Contrastive Learning for Unsupervised Textual Representations[C]. Proc. 59th Annual Meeting of the Association for Computational Linguistics and 11th International Joint Conference on Natural Language Processing (ACL/IJCNLP), Online, 2020: 879-895.

[25]Gao TY, Yao XC, Chen DQ. SimCSE: Simple Contrastive Learning of Sentence Embeddings[C]. Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP), Online, 2021: 6894-6910.

[26]Yan YM, Li RM, Wang SR, et al. ConSERT: A Contrastive Framework for Self-Supervised Sentence Representation Transfer[C]. Proc. 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL/IJCNLP), Online, 2021: 5065-5075.

[27]Peng H, Gao TY, Han X, et al. Learning from Context or Names? An Empirical Study on Neural Relation Extraction[C]. Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP), Online, 2020: 3661-3672.

[28]Qin YJ, Lin YK, Takanobu R, et al. ERICA: Improving Entity and Relation Understanding for Pre-trained Language Models via Contrastive Learning[C]. Proc. 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing

(ACL/IJCNLP), Online, 2021: 3350-3363.

[29]Su P, Peng YF, Vijay-Shanker K. Improving BERT Model Using Contrastive Learning for Biomedical Relation Extraction[C]. Proc. 20th Workshop on Biomedical Language Processing, Online, 2021: 1-10.

[30]Wei J, Zou K. EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks[C]. Proc. Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Hong Kong, China, 2019: 6382-6388.

[31]Chen T, Kornblith S, Norouzi M,et al. A simple framework for contrastive learning of visual representations[C]. Proc. 37th International Conference on Machine Learning (ICML), 2020: 1597-1607.

[32]Khosla P, Teterwak P, Wang C, et al. Supervised Contrastive Learning[C]. Proc. 34th Conference on Neural Information Processing Systems (NeurIPS), Vancouver, Canada, 2020.

[33]Bekoulis G, Deleu J, Demeester T, et al. Joint Entity Recognition and Relation Extraction as a Multi-head Selection Problem[J]. Expert Systems with Applications, 2018, 114: 34-45.

[34]Zheng SC, Wang F, Bao HY,et al. Joint Extraction of Entities and Relations Based on a Novel Tagging Scheme[C]. Proc. 55th Annual Meeting of the Association for Computational Linguistics (ACL), Vancouver, Canada, 2017: 1227-1236.

[35]Dan Roth, Wen-tau Yih. A Linear Programming Formulation for Global Inference in Natural Language Tasks [C]. Proc. 8th Conference on Computational Natural Language Learning (CoNLL-2004), Boston, Massachusetts, USA, 2004: 1-8.

[36]Luan Y, He LH, Ostendorf M, Hajishirzi H. Multi-Task Identification of Entities, Relations, and Coreference for Scientific Knowledge Graph Construction[C]. Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP), Brussels, Belgium, 2018: 3219-3232.

[37]Gurulingappa H, Rajput AM, Roberts A, et al. Development of a Benchmark Corpus to Support the Automatic Extraction of Drug-related Adverse Effects from Medical Case Reports [J]. Journal of Biomedical Informatics, 2012, 45(5):885-892.

[38]Giannis Bekoulis, Johannes Deleu, Thomas Demeester, et al. Adversarial training for multi-context joint entity and relation extraction[C]. Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP), Brussels, Belgium, 2018: 2830-2836.

[39]Makoto Miwa and Yutaka Sasaki. Modeling Joint Entity and Relation Extraction with Table Representation[C]. Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 2014: 1858-1869.

[40]Luan Y, He LH, Ostendorf M, Hajishirzi H. Multi-Task Identification of Entities, Relations, and Coreference for Scientific Knowledge Graph Construction[C]. Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP), Brussels, Belgium, 2018: 3219-3232.

[41]Wadden D, Wennberg U, Luan Y, Hajishirzi H. Entity, Relation, and Event Extraction with Contextualized Span Representations[C]. Proc. Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Hong Kong, China, 2019: 5784-5789.

[42]Li F, Zhang MS, Fu GH, Ji DH. A Neural Joint Model for Entity and Relation Extraction from Biomedical Text[J]. BMC Bioinformatics, 2017, 18(1): 1-11.