applied sciences

MDPI

Article

Joint Extraction of Multiple Relations and Entities from Building Code Clauses

Fulin Li [1], Yuanbin Song [1,*] and Yongwei Shan [2]

[1] School of Naval Architecture, Ocean & Civil Engineering, Shanghai Jiao Tong University, Shanghai 200240, China; fulinli@sjtu.edu.cn

[2] School of Civil and Environmental Engineering, Oklahoma State University, Stillwater, OK 74078, USA; yongwei.shan@okstate.edu

* Correspondence: ybsong@sjtu.edu.cn

Received: 6 September 2020; Accepted: 1 October 2020; Published: 13 October 2020

Abstract: The extraction of regulatory information is a prerequisite for automated code compliance checking. Although a number of machine learning models have been explored for extracting computer-understandable engineering constraints from code clauses written in natural language, most are inadequate to address the complexity of the semantic relations between named entities. In particular, the existence of two or more overlapping relations involving the same entity greatly exacerbates the difficulty of information extraction. In this paper, a joint extraction model is proposed to extract the relations among entities in the form of triplets. In the proposed model, a hybrid deep learning algorithm combined with a decomposition strategy is applied. First, all candidate subject entities are identified, and then, the associated object entities and predicate relations are simultaneously detected. In this way, multiple relations, especially overlapping relations, can be extracted. Furthermore, nonrelated pairs are excluded through the judicious recognition of subject entities. Moreover, a collection of domain-specific entity and relation types is investigated for model implementation. The experimental results indicate that the proposed model is promising for extracting multiple relations and entities from building codes.

Keywords: regulation information extraction; building code; deep learning; automated code compliance checking

1. Introduction

Automated code compliance checking is a promising alternative to manual checking that is expected to reduce time consumption, cost, and errors [1]. Thus, this approach has received extensive attention in the architectural, engineering, and construction industries for a number of applications. Some recent examples of such efforts include construction site layout assessment [2], green building evaluation [3], design-for-safety review [4], and building environmental monitoring [5]. For example, e-PlanCheck in CORENET [6] automatically check electronic building plans against engineering constraint, manually extracted from Singapore building codes. A typical case is to automatically review the area of a kitchen in a residential building, where the code requirement "The area of a kitchen shall not be less than 4 m^2" should be satisfied. Such an engineering constraint is hard-coded by software engineers into e-PlanCheck while the area of the kitchen is automatically calculated from a digital drawing. Information extraction is often a crucial linkage between the natural language and machine-readable language. In particular, named entities, such as "kitchen", "area", and "4 m^2" should be extracted from the aforementioned clause for kitchen design, and the semantic relations, like "has property" between "kitchen" and "area" as well as "not less than" between "area" and "4 m^2", should be extracted too. Automatic information extraction can save tremendous both time and cost

for knowledge acquisition [7]. Therefore, numerous approaches have been proposed to automate the extraction of regulation information.

A number of approaches applied a set of manually developed extraction rules to extract engineering constraints from code documents. Zhang and El-Gohary [8,9] utilized a rule-based extraction method with the assistance of natural language processing (NLP) techniques. Later, Zhou and El-Gohary [10] further developed a cascaded extraction algorithm to address more complex linguistic structures. Li et al. [11] proposed a specification language model for extracting spatial configuration constraints [12]. In these studies, experts must direct considerable effort and time toward the manual development of the extraction rules, which are often purpose-dependent rules with limited reusability. Moreover, conflicts may exist between some of the extraction rules in such a collection, and there are few systematic approaches available for identifying and resolving such rule conflicts in advance.

To overcome the aforementioned limitations, machine learning approaches have been explored for the extraction of regulatory information. They aim to automatically capture the underlying patterns of the text data, by learning from a large size of text data. Zhang and El-Gohary [13] adopted a conditional random field (CRF) algorithm to label the semantic roles of entities in building code sentences. Xu and Cai [14] used the labeled data in the FrameNet database [15] to train a probability model for extracting constraints in the form of semantic frames. However, most of these machine learning algorithms rely on shallow learning; consequently, extensive manual intervention and supervised NLP toolkits are still required to construct sophisticated features. As an emerging alternative, deep learning techniques have recently demonstrated a promising capacity for automatic feature engineering and have shown competitive performance for information extraction in fields such as medical text analysis [16] and social media mining [17]. In addition, some pioneering research has been conducted to explore the potential of deep learning techniques for improving the extraction of engineering constraints. Song et al. [18] used the word2vec model [19] to learn the semantics of both words and sentences in building codes. Later, the word2vec model was further integrated with a bidirectional long short-term memory (Bi-LSTM) model based on a CRF algorithm for identifying the key elements of semantic roles [20]. Zhong et al. [21] designed a pipeline combining named entity recognition (NER) with sentence classification for the extraction of typical temporal relationships between two construction activities (named entities).

Undoubtedly, these machine-learning-based methods offer a promising approach for the automatic extraction of regulatory information without the need for handcrafted rules. However, most of these studies have treated regulation information extraction as a process of NER and have inadequately addressed the complexity of extracting the semantic relations between entities. For instance, Zhong et al. [21] attempted to infer relations through sentence type classification. However, this paradigm is confined to the extraction of one relation from one simple clause involving two entities. By contrast, in practice, many Chinese code clauses describing engineering constraints are written in natural language and express two or more relations among a set of entities. In particular, due to the concise nature of code language, the entities in code clauses tend to appear at a high density. Moreover, overlapping relations often exist, meaning that multiple relations involve a common entity (herein called the "overlapping relation" problem).

Recent research has explored the joint extraction of entities and relations, which has shed some light on the automatic extraction of multiple relations from code clauses with complex structures. In contrast to the separate NER and relation classification tasks, the aim of joint extraction is to simultaneously detect multiple entities along with their relations using a single model [22]. Unfortunately, the current joint extraction models are not suitable for analyzing code documents since the entities and relations used for labeling documents and training the models are often domain-specific and purpose-dependent. Furthermore, overlapping relations, which frequently exist in code documents, exacerbate the difficulty of information extraction. Therefore, the main purpose of this paper is to extract complex semantic relationships residing in code documents. For this

purpose, a joint extraction model is developed based on the application of a decomposition strategy. Meanwhile, a collection of domain-specific entity and relation types is proposed from the viewpoint of the ontology of building design. Compared with the existing machine-learning-based regulation information extraction, this research contributes to the body of knowledge by providing a joint learning approach to enhance the acquisition of complex semantic relations in terms of multiple relations and overlapping relations. Remarkably, this paper attempts to better utilize NLP techniques with the assistance of the ontological description of semantic relations and named entities in the context of regulation information extraction.

The remaining of this paper is organized as follows: Section 2 presents the semantic types of named entities and relations while the framework of the joint extraction model is proposed with algorithm description of each module. In Section 3, the experiment process and the hardware platform are introduced, and later the resultant data are evaluated to validate the performance of the proposed model. Finally, the conclusions and limitations are summarized in Section 4.

2. Methodology

2.1. Semantic Types of Named Entities and Relations

In knowledge engineering, a binary relation between named entities is often represented by a triplet, where the predicate defines the specific relation (directional arc) and the subject and object denote the associated entities (node). Such a triplet is often called a subject-predicate-object (SPO) triplet [23]. Due to the expressiveness and flexibility of SPO triplets, they have been widely used to capture regulatory knowledge in the construction industry [24,25]. Accordingly, the triplet representation can also be used to represent various types of constraints expressed in building codes. In general, a named entity in a code clause can be a real-world building object, such as a building, a built space, or a construction element (see Table 1). It can also be an engineering property, a quantitative value, or a feature representing the function or purpose of an object.

Table 1. Named entity categories.

Entity Category	Label	Example
Building	E_1	residential building
Built space	E_2	kitchen
Construction element	E_3	wall
Feature	E_4	accessible
Property	E_5	height
Quantity	E_6	4.0 m

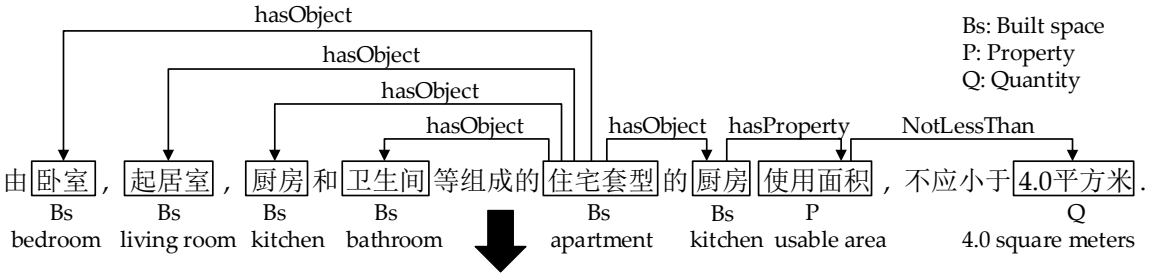
The predicates in code clauses can describe the system hierarchy in terms of the relations between high-level building objects (in the subject role) and low-level building objects (in the object role); see Table 2. In addition, an engineering property associated with a particular building object can be represented by the “hasProperty” predicate. Likewise, a function or purpose realized by a building object can be represented by the “hasFeature” predicate. Various types of spatial relationships can also be modeled in the form of predicates such as “AccessTo”, “Within”, “Outside”, “Between”, and “AdjacentTo”. Value constraints can be represented by comparative predicates, for example, “LessThan”, “GreaterThan”, and “EqualTo”. In addition, a multiplicative factor for a value obtained by referencing another property can be similarly defined by the “MultipliedBy” predicate.

Table 2. Predicate relation categories and the alternative semantic roles of associated entities.

Relation Category	Predicate	Label	Semantic Role		Example
			Subject	Object	
System hierarchy	hasObject	R_1	E_1	E_1,E_3	[building, hasObject, wall]
			E_2	E_2,E_3	[toilet, hasObject, window]
			E_3	E_3	[stair, hasObject, step]
Engineering property	hasProperty	R_2	E_1,E_2,E_3	E_5	[kitchen, hasProperty, area]
Function and purpose	hasFeature	R_3	E_1,E_2,E_3	E_4	[entrance, hasFeature, accessible]
Spatial relationship	Within	R_4	E_1,E_2,E_3	E_1,E_2	[pipeline, Within, building]
	Outside	R_5	E_1,E_2,E_3	E_1,E_2	[fire hydrant, Outside, tunnel]
	Between	R_6	E_1,E_2,E_3,E_5	E_1,E_2,E_3	[clearance, Between, wall]
	AdjacentTo	R_7	E_1,E_2,E_3	E_1,E_2,E_3	[road, AdjacentTo, entrance]
	AccessTo	R_8	E_1,E_2,E_3	E_1,E_2,E_3	[corridor, AccessTo, bedroom]
Comparative relation	NotLessThan	R_9	E_5	E_5,E_6	[height, NotLessThan, 2 m]
	NotGreaterThan	R_{10}	E_5	E_5,E_6	[height, NotGreaterThan, 2 m]
	LessThan	R_{11}	E_5	E_5,E_6	[height, LessThan, 2 m]
	GreaterThan	R_{12}	E_5	E_5,E_6	[height, GreaterThan, 2 m]
	EqualTo	R_{13}	E_5	E_5,E_6	[height, EqualTo, 2 m]
Quantity reference	MultipliedBy	R_{14}	E_5	E_6	[area, MultipliedBy, 1/2]

The fourth and fifth columns of Table 2 specify the alternative semantic roles of the two entities associated with a particular predicate. In particular, physical buildings/elements are distinguished from built (functional) spaces, as specified in ISO 12006-2 [26]. For example, construction elements such as walls, slabs, doors, and windows can constitute a physical building and can also systematically function as a living space. In this sense, a physical building (labeled E_1) can contain low-level building system (labeled E_1) and/or construction elements (labeled E_3), as shown in the first row of Table 2. This distinction can facilitate the development of more accurate labeling rules. Unfortunately, general information extraction research provides very little guidance for defining the semantic roles of the subject and object entities associated with the domain predicates listed in Table 2.

Figure 1 shows an example of multiple triplets in a Chinese building code clause: “由卧室, 起居室, 厨房和卫生间等组成的住宅套型的厨房使用面积, 不应小于4.0平方米 (The usable area of a kitchen in an apartment consisting of a bedroom, living room, kitchen, and bathroom shall be no less than 4.0 square meters)”. In this sentence, eight entities and seven relations exist, constituting seven triplets. The five “hasObject” predicates linking the subject “住宅套型#1 (apartment)” to five built spaces (object entities) indicate the systematic organization of the apartment. The “hasProperty” predicate describes the space-property relationship between the subject entity “厨房#2 (kitchen)” and the object entity “使用面积#1 (usable area)”. The “NotLessThan” predicate describes a lower-value constraint (“4.0 meters”) on the “使用面积#1 (usable area)”. Some entities, for example, “厨房 (kitchen)” in Figure 1, may appear more than once in the same sentence. The two instances of this entity are further labeled #1 and #2 in accordance with the order of their appearance.



Extracted Triplets

Subject entity	Predicate relation	Object entity
"住宅套型"#1(18,21) (apartment)	hasObject	"卧室"#1(1,2) (bedroom)
"住宅套型"#1(18,21) (apartment)	hasObject	"起居室"#1(4,6) (living room)
"住宅套型"#1(18,21) (apartment)	hasObject	"厨房"#1(8,9) (kitchen)
"住宅套型"#1(18,21) (apartment)	hasObject	"卫生间"#1(11,13) (bathroom)
"住宅套型"#1(18,21) (apartment)	hasObject	"厨房"#2(23,24) (kitchen)
"厨房"#2(23,24) (kitchen)	hasProperty	"使用面积"#1(25,28) (usable area)
"使用面积"#1(25,28) (usable area)	NotLessThan	"4.0平方米"#1(34,39) (4.0 square meters)

Format: ["entity" #index (start position, end position)]

Figure 1. Example illustrating multiple triplets in a clause.

2.2. Joint Extraction Model

Currently, a number of models have been developed for the joint extraction of entities and relations [27–29]. However, few models excel at solving the overlapping relation problem. For example, in Figure 1, eight entities exist in the clause, and five different relations refer to the common entity "住宅套型#1 (apartment)". To account for situations of this kind, some models assume that potential relations exist between each pair of identified entities [30]. In general, N entities can produce approximately N^2 candidate pairs; however, most of these are meaningless, resulting in low relation extraction performance, especially when entities are densely present in the sentences of building codes. For instance, fifty clauses from Chinese building codes were randomly selected and labeled using the predefined entity and relation types presented in Section 2.1 to investigate the existence of overlapping relations in each sample clause. On average, 6.1 entities and 4.9 relations were identified per code clause. These results show that compared with documents from other domains (such as the dataset of New York Times articles, which contains, on average, 3.3 entities and 1.7 relations per sentence), entities appear more densely in building codes, and the relations between these entities are richer. Moreover, the overlapping relation problem is exacerbated by the fact that although each relation has only one subject entity, a single subject entity may be involved in several relations.

To address the aforementioned overlapping relation problem, a joint extraction model is developed based on the application of a decomposition strategy. In the proposed model, all subject entities are first identified using a hybrid deep learning algorithm, and subsequently, for each identified subject entity, its associated object entities and predicate relations are simultaneously extracted. In this way, nonrelated pairs can be excluded through the judicious recognition of subject entities, and the overlapping relation problem can be naturally addressed.

2.2.1. Model Framework

Figure 2 illustrates the framework of the joint extraction model. Following the pipeline of triplets extraction model [22], four components, that is, augmented character embedding, a shared semantic encoder, a subject extractor, and an object and predicate extractor are specifically developed. The first module, augmented character embedding, transforms the sequence of discrete characters of a raw

code clause into vectors of real numbers, herein called character vectors. Subsequently, a shared semantic encoder is applied to learn the context features of each character, literally called task-shared features [31], across different tasks via parameter sharing mechanism. Then, the subject extractor uses the task-shared features to identify all potential entities that can act as subjects in predicate relations. For each identified subject entity, its associated object entities and predicate relations are then simultaneously identified by the object and predicate extractor. Finally, the initial raw code clause is automatically transformed into a set of triplets that can be parsed by computers.

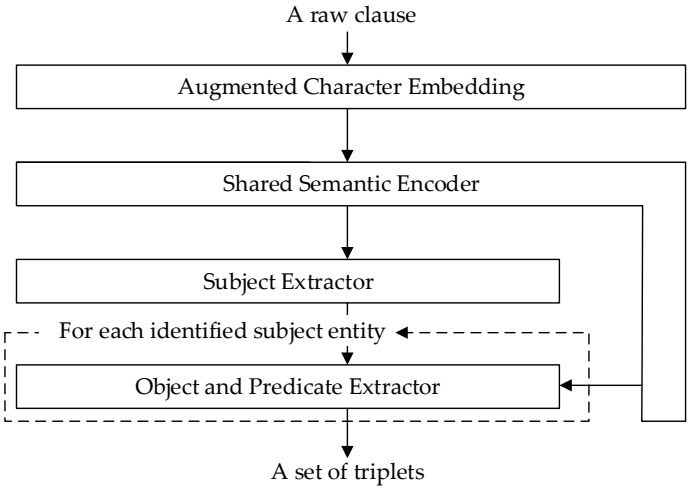


Figure 2. Framework of the joint extraction model.

2.2.2. Augmented Character Embedding

Word embedding is a technique for representing characters/words as real-valued vectors of high density and low dimensionality. Vectors representing a collection of words or characters with similar semantic meanings exhibit high cosine similarity. For example, the cosine value of the angle between the vector for "door" and that for "window" is 0.72, indicating high semantic similarity. In this way, the sequence of discrete characters forming a sentence or even a paragraph can be transformed into a numeric representation for further processing by deep learning algorithms.

Unlike written English, written Chinese does not use delimiters between words. Incorrect word segmentation may lead to erroneous entity recognition and information extraction failure. Therefore, for pragmatic reasons, each Chinese character is frequently treated as an atomic linguistic unit. However, one drawback of general character embedding is the lack of word information expressed, especially in the context of building code analysis, which involves dozens of characteristic technical terms. Thus, an augmented embedding method is designed to enrich Chinese characters with word information (see Figure 3, Part 1). Specifically, the Building Information Model Classification and Coding Standard (GB/T 51269-2017) [32] is utilized as the terminology reference for deriving the prior semantic information that needs to be represented by word-level embedding vectors. These word-level vectors can then be integrated into character-level vectors. In this way, the resulting vectors can express characters with richer semantics.

Formally, for a given clause $S = \{c_1, c_2, \ldots, c_n\}$, where c_i represents the i-th character in S, the input representation for each character $\boldsymbol{x}_i \in \mathbb{R}^{d_c}$ is as follows:

$$\boldsymbol{x}_i = \left[\mathbf{e}^c\left(c_i\right) + \boldsymbol{W}_T\mathbf{e}^w\left(\mathbf{word}\left(c_i\right)\right)\right], \tag{1}$$

where $\mathbf{e}^c$ and $\mathbf{e}^w$ denote a pretrained character embedding lookup table and a pretrained word embedding lookup table, respectively. The formula $\mathbf{word}\left(c_i\right)$ represents the word to which character c_i belongs, which is implemented based on the Jieba segmentation library. $\boldsymbol{W}_T \in \mathbb{R}^{d_c \times d_w}$ is a trainable transformation matrix for aligning the dimensionality of a word vector (d_w) with that of a character

vector (d_c). Finally, the sequence of characters that forms a raw clause can be expressed as a vector matrix, where each column represents the character vector $\boldsymbol{x}_i$ associated with the corresponding position in the sentence.

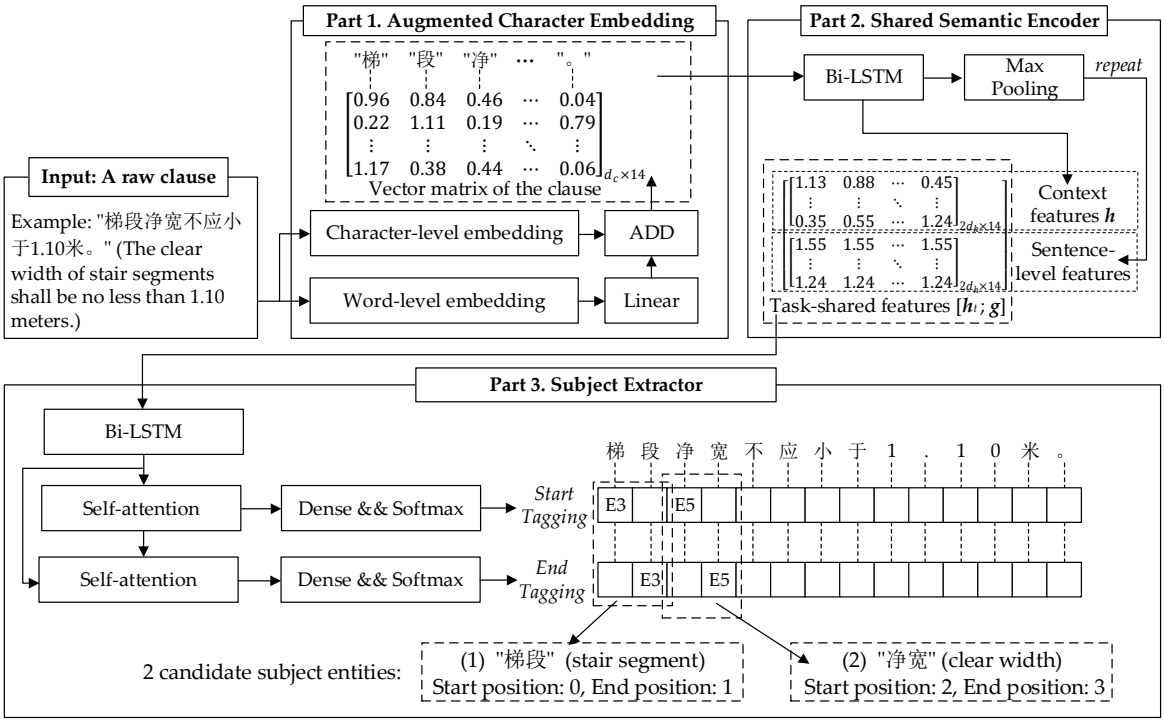


Figure 3. The extraction of candidate subject entities. In this figure, the clause "The clear width of stair segments shall be no less than 1.10 m (in Chinese: 梯段净宽不应小于1.10米"), with a length of n = 14, is taken as an example.

2.2.3. Shared Semantic Encoder

The augmented character embedding vectors provide only prior semantic information for each character in a clause. A character/word will also convey more specific information depending on how it is used in a sentence. The sequence of characters in a sentence forms the context of those characters. To further improve the analysis capabilities, this context information can also be embedded into the character embedding vectors. Accordingly, a shared semantic encoder (see Figure 3, Part 2) is designed to utilize a Bi-LSTM network to capture the context features for each character. In detail, in the forward direction, the Bi-LSTM network computes along the sequence of input vectors from left to right, while in the backward direction, it computes from right to left. An LSTM model includes gate mechanisms to "remember" (obtain new information) and "forget" (discard unnecessary information) [33]. In this way, the association between a character and its surrounding characters (on both the left- and right-hand sides) is encoded by concatenating the forward and backward LSTM hidden states. The output of a Bi-LSTM layer is as follows:

$$\boldsymbol{h}_t = \text{BiLSTM}(\boldsymbol{x}_t) = [\overrightarrow{\boldsymbol{h}_t} \oplus \overleftarrow{\boldsymbol{h}_t}], \tag{2}$$

where $\overrightarrow{\boldsymbol{h}_t}$ and $\overleftarrow{\boldsymbol{h}_t}$ are the forward and backward hidden states, respectively, at position t and $\oplus$ represents the concatenation operator. The function of BiLSTM is directly imported from the TensorFlow library.

Moreover, in addition to the associations with adjacent characters within a certain "distance", a sentence-level feature $\boldsymbol{g}$ is computed via max pooling over all hidden states of the Bi-LSTM layer. Finally, each hidden state $\boldsymbol{h}_t$ is concatenated with $\boldsymbol{g}$ to form the task-shared feature vectors $[\boldsymbol{h}_t; \boldsymbol{g}]$.

These vectors are input into two downstream processes, that is, the subject extractor and the object and predicate extractor.

2.2.4. Subject Extractor

The subject extractor is responsible for perceiving all candidate subject entities for the target SPO triplets. In Chinese code clauses, a domain-specific named entity is frequently composed of multiple characters. For example, the named entity "变压器", comprising three characters, means "transformer". To identify such entities, we first identify the start positions of the candidate subject entities and then identify their end positions. Specifically, two tagging processes are employed: one for the start position and the other for the end position. When a character is tagged as a start or end position, its corresponding entity type is also labeled. In addition, the start position tagging information is used as input to the end position tagging process to achieve more accurate predictions.

Figure 3, Part 3, illustrates the processing procedure of the subject extractor. First, a Bi-LSTM model is used to extract task-specific feature vectors from the task-shared feature vectors. Subsequently, a multihead self-attention model is applied to the feature vectors to tag the start position of each subject entity along with its entity type. Self-attention is a method of encoding sequences of vectors by relating these vectors to each other based on pairwise similarities [34]. By means of self-attention mechanism, a model can determine which characters are important for specific tasks. Essentially, an attention function is a mapping function that maps a query vector $\boldsymbol{Q}$ and a set of key-value vector pairs ($\boldsymbol{K}$, $\boldsymbol{V}$) to an output. In this case, the scaled dot-product attention is adopted (see Equation (3)). The query vector of a character is applied to the key vector of each other character in the sentence and then scaled by the dimensionality of the hidden units. Subsequently, a softmax function is utilized to compute the pairwise similarity between the query character and the key features of each other character of the sentence. The softmax is a probabilistic function that takes real numbered inputs and outputs a probability number. This function is directly imported from the TensorFlow library. Finally, the attention result can be calculated by taking the dot product of the softmax output and the value vector of each character.

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{Softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{\boldsymbol{d}}}\right)\boldsymbol{V}, \tag{3}$$

where $\boldsymbol{Q} \in \mathbb{R}^{n\times 2d_h}$, $\boldsymbol{K} \in \mathbb{R}^{n\times 2d_h}$, and $\boldsymbol{V} \in \mathbb{R}^{n\times 2d_h}$ are the query matrix, key matrix and value matrix, respectively. $\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})$ is the final attention value calculated by pairwise similarity scores and value vector. $\boldsymbol{h}^{se} = [\boldsymbol{h}_1^{se}, \boldsymbol{h}_2^{se}, \ldots, \boldsymbol{h}_n^{se}]$ denotes the task-specific features output by the Bi-LSTM layer. In this paper, $\boldsymbol{Q} = \boldsymbol{K} = \boldsymbol{V} = \boldsymbol{h}^{se}$. $\boldsymbol{d}$ is the dimensionality of the hidden units of the Bi-LSTM layer, which is equal to $2d_h$.

Furthermore, the multihead attention model employs multiple different linear layers to linearly map vectors $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ and then focuses on obtaining the multihead attention. The intuition behind the multihead attention is that applying the attention multiple times may learn more abundant features than single attention (also known as a head) in the sentence [34]. If the multihead attention contains m heads, the i-th attention head can be calculated using Equation (4). And the final multihead attention is the concatenation of all heads as shown in Equation (5).

$$head_i = \text{Attention}(\boldsymbol{Q}\boldsymbol{W}_i^Q, \boldsymbol{K}\boldsymbol{W}_i^K, \boldsymbol{V}\boldsymbol{W}_i^V) \tag{4}$$

$$\text{MultiHead}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = (head_1 \oplus \ldots \oplus head_m)\boldsymbol{W}_O, \tag{5}$$

where $\boldsymbol{W}_i^Q \in \mathbb{R}^{2d_h\times d_k}$, $\boldsymbol{W}_i^K \in \mathbb{R}^{2d_h\times d_k}$, and $\boldsymbol{W}_i^V \in \mathbb{R}^{2d_h\times d_k}$ are trainable projection parameters. $d_k = 2d_h/m$, and $\boldsymbol{W}_O \in \mathbb{R}^{2d_h\times 2d_h}$ is also a trainable parameter. $\text{MultiHead}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})$ is the final multihead attention value. The function of $\text{MultiHead}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})$ is implemented based on the TensorFlow library according to the work of Vaswani et al. [34]. Subsequently, the output of the

self-attention layer $\boldsymbol{h}^{se-sta} = \left[\boldsymbol{h}_1^{se-sta}, \boldsymbol{h}_2^{se-sta}, \ldots, \boldsymbol{h}_n^{se-sta}\right]$ is fed to a dense layer with softmax activation to produce a probability distribution over all labels. The label of character c_t when it is tagged as a start position is predicted as shown in Equation (7):

$$P\left(y_t^{se-sta}\right) = \text{Softmax}\left(\boldsymbol{W}^{se-sta}\boldsymbol{h}_t^{se-sta} + \boldsymbol{b}^{se-sta}\right) \tag{6}$$

$$\text{SE-TAG}_{\text{sta}}\left(\boldsymbol{x}_t\right) = \arg\max_k P\left(y_t^{se-sta} = k\right), \tag{7}$$

where $\boldsymbol{W}^{se-sta} \in \mathbb{R}^{|T|\times 2d_h}$ and $\boldsymbol{b}^{se-sta} \in \mathbb{R}^{|T|}$ are trainable parameters. $|T|$ denotes the number of output labels.

Likewise, the multihead self-attention model can be adopted to capture the features for subject-entity end position identification. Considering that the prediction of the end positions may benefit from the prediction results for the start positions, the output of the first self-attention layer, $\boldsymbol{h}^{se-sta}$, is concatenated with the task-specific features $\boldsymbol{h}^{se}$ to initialize $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$ in the second self-attention layer. The output of the second layer is denoted by $\boldsymbol{h}^{se-end} = \left[\boldsymbol{h}_1^{se-end}, \boldsymbol{h}_2^{se-end}, \ldots, \boldsymbol{h}_n^{se-end}\right]$. Then, the label of character c_t when it is tagged as an end position is predicted as shown in Equation (9):

$$P\left(y_t^{se-end}\right) = \text{Softmax}\left(\boldsymbol{W}^{se-end}\boldsymbol{h}_t^{se-end} + \boldsymbol{b}^{se-end}\right) \tag{8}$$

$$\text{SE-TAG}_{\text{end}}\left(\boldsymbol{x}_t\right) = \arg\max_k P\left(y_t^{se-end} = k\right), \tag{9}$$

where $\boldsymbol{W}^{se-end} \in \mathbb{R}^{|T|\times 4d_h}$ and $\boldsymbol{b}^{se-end} \in \mathbb{R}^{|T|}$ are trainable parameters.

Figure 3 shows an example of subject-entity extraction. The candidate subject entities are each tagged with the corresponding entity type label ("E3" for "construction element" and "E5" for "property") in the start and end tag sequences. Accordingly, a set of subject-entities (i.e., "梯段" (stair segment) and "净宽" (clear width)) are extracted. In addition, the training loss of the subject extractor (which is to be minimized) is defined as the sum of the negative log probabilities of the true start and end tags based on the predicted distributions:

$$\mathcal{L}_{se} = -\frac{1}{n}\sum_{i=1}^{n}\left(\log P\left(y_i^{se-sta} = \hat{y}_i^{se-sta}\right) + \log P\left(y_i^{se-end} = \hat{y}_i^{se-end}\right)\right), \tag{10}$$

where $\hat{y}_i^{se-sta}$ and $\hat{y}_i^{se-end}$ are the true start and end tags, respectively, of the i-th character, and n is the length of the input clause.

2.2.5. Object and Predicate Extractor

Following the extraction of a set of candidate subject entities, their corresponding object entities and the predicates used to link a subject and an object are simultaneously derived by the object and predicate extractor (see Figure 4). The semantic information of a subject entity is helpful for predicting the corresponding object entities and predicate relations. For a given subject entity composed of a sequence of characters, its semantic information can be encoded using an LSTM model, character by character, to further enrich the task-shared feature vectors acquired from the shared semantic encoder. Specifically, the task-shared features for each character in a subject entity are sequentially input into the LSTM model. Finally, the last hidden states of the LSTM are used as the semantic information of the given subject entity and are concatenated with the task-shared features. The resulting concatenated vector contains both context-enriched character information and the associated subject-entity information. Subsequently, two tagging processes are sequentially performed to identify the start and end positions of each object entity. Moreover, whenever a character is tagged as a start position, the associated predicate relation type is simultaneously labeled.

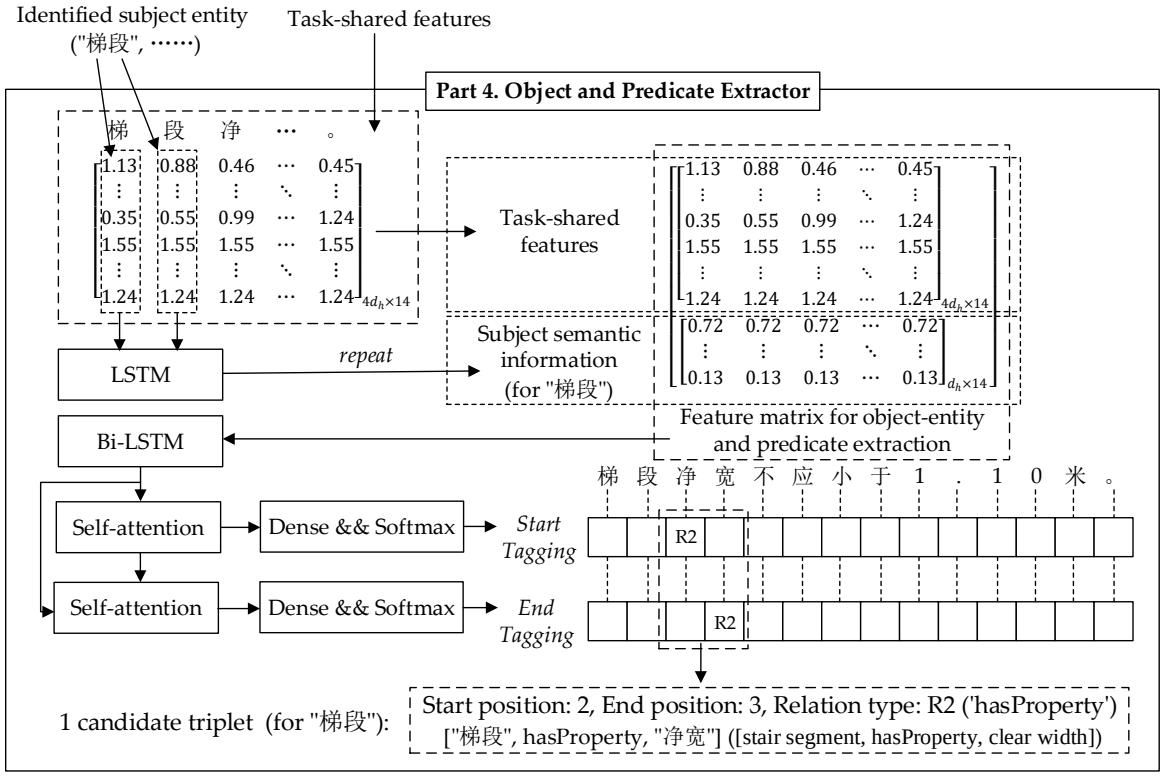


Figure 4. The extraction of object entities and predicate relations. In this figure, the identified subject entity "梯段" (stair segment) is taken as an example.

The identification of the start and end positions for the object entities is very similar to that for the subject entities, except that the tags for both the start and end positions of an object entity indicate the type of the predicate relation rather than the type of the object entity. For example, in Figure 4, when the given subject entity is "梯段 (stair segment)", the start and end positions of the entity "净宽" (clear width) are tagged with the label of the predicate type "R2" (i.e., "hasProperty") in the start and end tag sequences, respectively. Accordingly, one candidate triplet for the given subject entity "梯段 (stair segment)" is extracted, that is, ["梯段", hasProperty, "净宽"] ([stair segment, hasProperty, clear width]). In this way, for each identified subject entity, its associated object entities and predicate relations are simultaneously extracted.

Formally, given an identified subject entity, the label of character c_t when it is tagged as a start or end position is predicted as shown in Equation (12) and Equation (14), respectively:

$$P\left(y_t^{ope-sta}\right) = \text{Softmax}\left(\boldsymbol{W}^{ope-sta}\boldsymbol{h}_t^{ope-sta} + \boldsymbol{b}^{ope-sta}\right) \tag{11}$$

$$\text{OPE-TAG}_{\text{sta}}\left(\boldsymbol{x}_t\right) = \arg\max_k P\left(y_t^{ope-sta} = k\right) \tag{12}$$

$$P\left(y_t^{ope-end}\right) = \text{Softmax}\left(\boldsymbol{W}^{ope-end}\boldsymbol{h}_t^{ope-end} + \boldsymbol{b}^{ope-end}\right) \tag{13}$$

$$\text{OPE-TAG}_{\text{end}}\left(\boldsymbol{x}_t\right) = \arg\max_k P\left(y_t^{ope-end} = k\right), \tag{14}$$

where $\boldsymbol{h}_t^{ope-sta}$ and $\boldsymbol{h}_t^{ope-end}$ are the outputs of the first and second self-attention layers, respectively, at position t and $\boldsymbol{W}^{ope-sta} \in \mathbb{R}^{|T'|\times 2d_h}$, $\boldsymbol{W}^{ope-end} \in \mathbb{R}^{|T'|\times 4d_h}$, and $\boldsymbol{b}^{ope-sta}, \boldsymbol{b}^{ope-end} \in \mathbb{R}^{|T'|}$ are trainable parameters. $|T'|$ denotes the number of output labels. Finally, the training loss of the object and

predicate extractor is defined as the sum of the negative log probabilities of the true start and end tags based on the predicted distributions:

$$\mathcal{L}_{ope} = -\frac{1}{n}\sum_{i=1}^{n}\left(\log P\left(y_i^{ope-sta} = \hat{y}_i^{ope-sta}\right) + \log P\left(y_i^{ope-end} = \hat{y}_i^{ope-end}\right)\right), \tag{15}$$

where $\hat{y}_i^{ope-sta}$ and $\hat{y}_i^{ope-end}$ are the true start and end tags, respectively, of the i-th character and n is the length of the input clause.

2.3. Model Training and Inference

In the training phase, the subject extractor and the object and predicate extractor can be jointly trained since they share the same input of the code clauses. For each training instance, a subject entity is randomly selected from the training data set of subject entities as the specified input to the object and predicate extractor. By comparing the model output with the actual labeled results, the loss functions of the subject extractor and the object and predicate extractor can be calculated; these losses are then combined to calculate the total loss, as follows:

$$\mathcal{L} = \mathcal{L}_{se} + \mathcal{L}_{ope}. \tag{16}$$

The total loss function in Equation (16) can be optimized using the Adam algorithm [35] to consider the mutual influence of the errors arising in the extraction of the subject entities, object entities, and predicates such that the errors that occur in each task are constrained by the others.

Algorithm 1 Inference Algorithm.

Input:
S
S denotes the input clause
Output:
$\{< s_i, p_{ij}, o_{ij} >\}$
s_i denotes the i-th extracted subject entity; p_{ij} and o_{ij} are the j-th predicate relation and object entity, respectively, associated with the i-th subject entity.

Define $n \leftarrow$ clause length
Initialize $\mathcal{T} \leftarrow \{\}$
Initialize $\mathcal{S} \leftarrow \{\}$
Obtain SE-TAG$_{\text{sta}}(S)$ using Equation (7)
Obtain SE-TAG$_{\text{end}}(S)$ using Equation (9)
for $i \leftarrow 1$ to n **do**
    if SE-TAG$_{\text{sta}}(S)[i] \neq$ "O" **then**
        for $j \leftarrow i$ to n **do**
            if SE-TAG$_{\text{sta}}(S)[i] =$ SE-TAG$_{\text{sta}}(S)[j]$ **then**
                $s \leftarrow S[i:j]$
                $\mathcal{S} \leftarrow \mathcal{S} \cup s$
                Break
for $s^* \leftarrow s$ in $\mathcal{S}$ **do**
    Obtain OPE-TAG$_{\text{sta}}(S)$ using Equation (12)
    Obtain OPE-TAG$_{\text{end}}(S)$ using Equation (14)
    for $i \leftarrow 1$ to n **do**
        if OPE-TAG$_{\text{sta}}(S)[i] \neq$ "O" **then**
            for $j \leftarrow i$ to n **do**
                if OPE-TAG$_{\text{sta}}(S)[i] =$ OPE-TAG$_{\text{end}}(S)[j]$ **then**
                $o \leftarrow S[i:j]$
                $p \leftarrow$ OPE-TAG$_{\text{sta}}(S)[i]$
                $t \leftarrow < s^*, p, o >$
                $\mathcal{T} \leftarrow \mathcal{T} \cup t$
                Break
return $\mathcal{T}$

Algorithm 1 presents the pseudocode for decoding the tagging results from the subject extractor and the object and predicate extractor. Lines 1–3 represent the initialization process for each input clause S expressed in natural language. Lines 4–12 describe the algorithm for deriving candidate subject entities from the tagging results of the subject extractor. Then, Lines 13–24 describe the algorithm for acquiring all candidate triplets from the tagging results of the object and predicate extractor. In detail, for a sentence with k subject entities, the entire task is finally deconstructed into two tasks for subject-entity tagging and $2k$ tasks for object-entity and predicate tagging. By means of this inference algorithm, the triplets can be directly identified from the start/end positions and tags of the entities, thus helping to accelerate the inference process and reduce the demand for computing resources. Moreover, nonrelated pairs can be excluded through the judicious recognition of the subject entities, and the overlapping relation problem can be naturally resolved.

3. Experiment

3.1. Data Preparation and Labeling

An experiment was conducted to validate the performance of the proposed joint extraction method on 1320 code clauses selected from 14 building codes written in Chinese (Table 3). Three domain experts were invited to annotate all triplet instances appearing in the code clauses based on the predefined types of entities and relations listed in Tables 1 and 2. First, text span annotations for the subject entities and object entities were labeled, and then, the tags of the predicates were determined. The manual labeling process was facilitated by BRAT, an intuitive web-based text labeling tool. Figure 5 presents eight typical examples of labeled sentences.

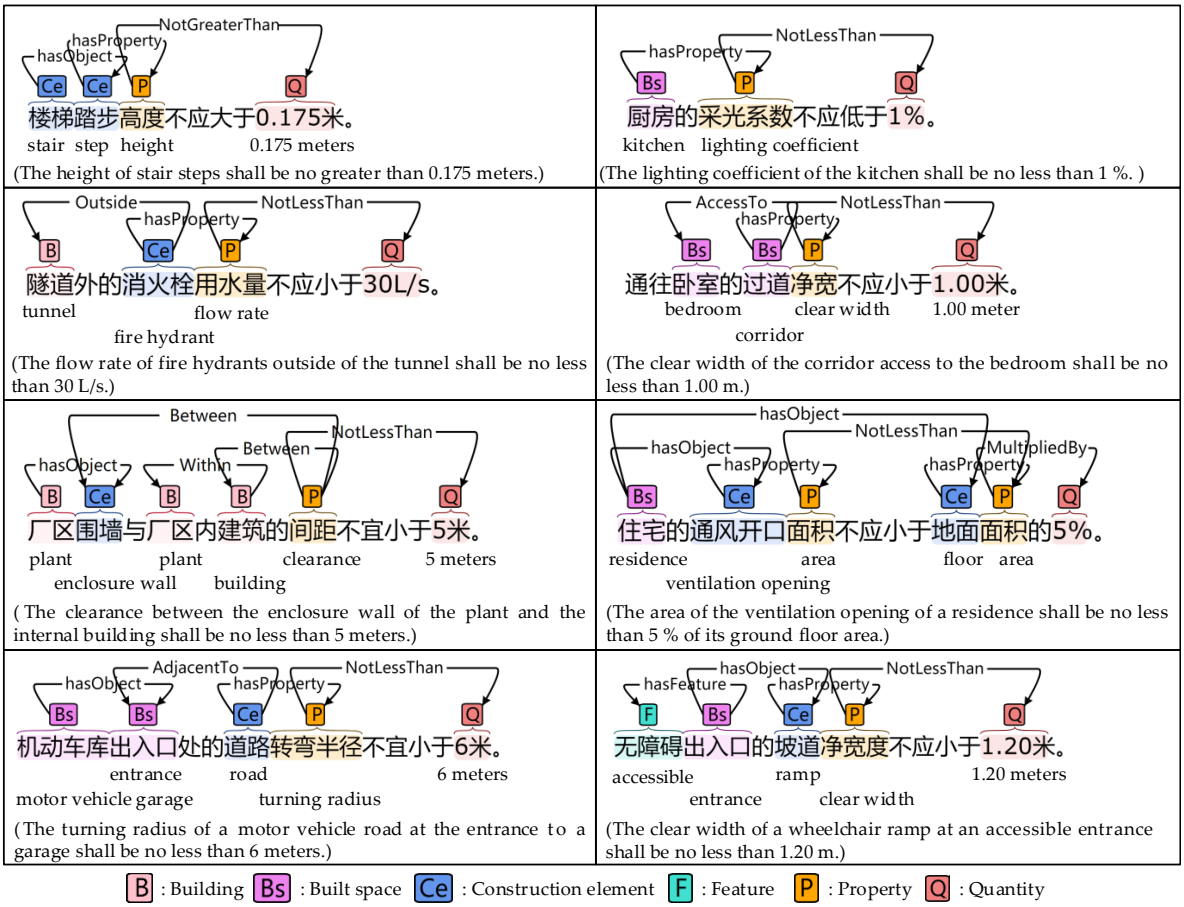


Figure 5. Labeled clauses for model training and testing (partial).

Table 3. Chinese building codes used for dataset preparation.

Standard No.	Name of Chinese Building Codes
GB50368-2005	《住宅建筑规范》(Residential building code)
GB50096-2011	《住宅设计规范》(Design code for residential buildings)
GB50352-2019	《民用建筑设计统一标准》(Uniform standard for the design of civil buildings)
GB50763-2012	《无障碍设计规范》(Code for accessibility design)
GB50016-2014	《建筑设计防火规范》(Code for the fire protection design of buildings)
GB50099-2011	《中小学校设计规范》(Code for the design of schools)
GB51039-2014	《综合医院建筑设计规范》(Code for the design of general hospitals)
GB50067-2014	《公共建筑节能设计标准》(Design standard for the energy efficiency of public buildings)
GB50038-2005	《人民防空地下室设计规范》(Code for the design of civil air defense basements)
JGJ100-2015	《车库建筑设计规范》(Code for the design of parking garage buildings)
JGJ450-2018	《老年人照料设施建筑设计标准》(Standard for the design of care facilities for the aged)
JGJ39-2016	《托儿所、幼儿园建筑设计规范》(Code for the design of nursery and kindergarten buildings)
JGJ48-2014	《商店建筑设计规范》(Code for the design of store buildings)
JGJ62-2014	《旅馆建筑设计规范》(Code for the design of hotel buildings)

3.2. Implementation of the Joint Extraction Model

A total of 924 clauses, 70% of the labeled clause set, were randomly selected as the training set, while the others were used for testing. The model was implemented using TensorFlow on a PC equipped with a 4.00 GHz Intel(R)i7-6700K CPU, two NVIDIA GeForce GTX 1080 GPUs and 64 GB of RAM. The dimensionality of the LSTM hidden states was 128. Dropout was applied to the embeddings and hidden states at a rate of 0.3. The number of attention heads was set to 8. The character and word embeddings used in this experiment were pretrained on the Baidu Encyclopedia corpus using the word2vec toolkit [19], and the number of embedding dimensions was set to 300. The model parameters were optimized for 500 iterations using Adam [35] with a learning rate of 1e-4 and a batch size of 64.

For comparison, the following classic triplet extraction models were employed as baselines: (1) Ren et al. [36] proposed a domain-independent framework for jointly learning representations of entity mentions, relation mentions, and type labels; (2) Zheng et al. [37] converted the joint triplet extraction task into a tagging problem based on a unified tagging scheme; (3) Tan et al. [28] designed a joint extraction model based on a translation mechanism; (4) Zeng et al. [38] considered triplet extraction as a sequence-to-sequence problem with a copy mechanism; and (5) Fu et al. [30] first adopted graph convolutional networks to extract text features and then considered all word pairs for triplet prediction. The test results of the aforementioned models are compared with those of our model in the next section.

3.3. Experimental Results and Analysis

Figure 6 illustrates the test results of the trained joint extraction model. Specifically, the extracted triplets with green backgrounds are correct, whereas a red background indicates an incorrect prediction, and a yellow background indicates an omitted triplet. The test results imply that the model has the capacity to discover multiple relations residing in code clauses, especially overlapping relations. Moreover, the number of extracted triplets varies for many test sentences that cannot be processed using the extraction model proposed in Reference [21].

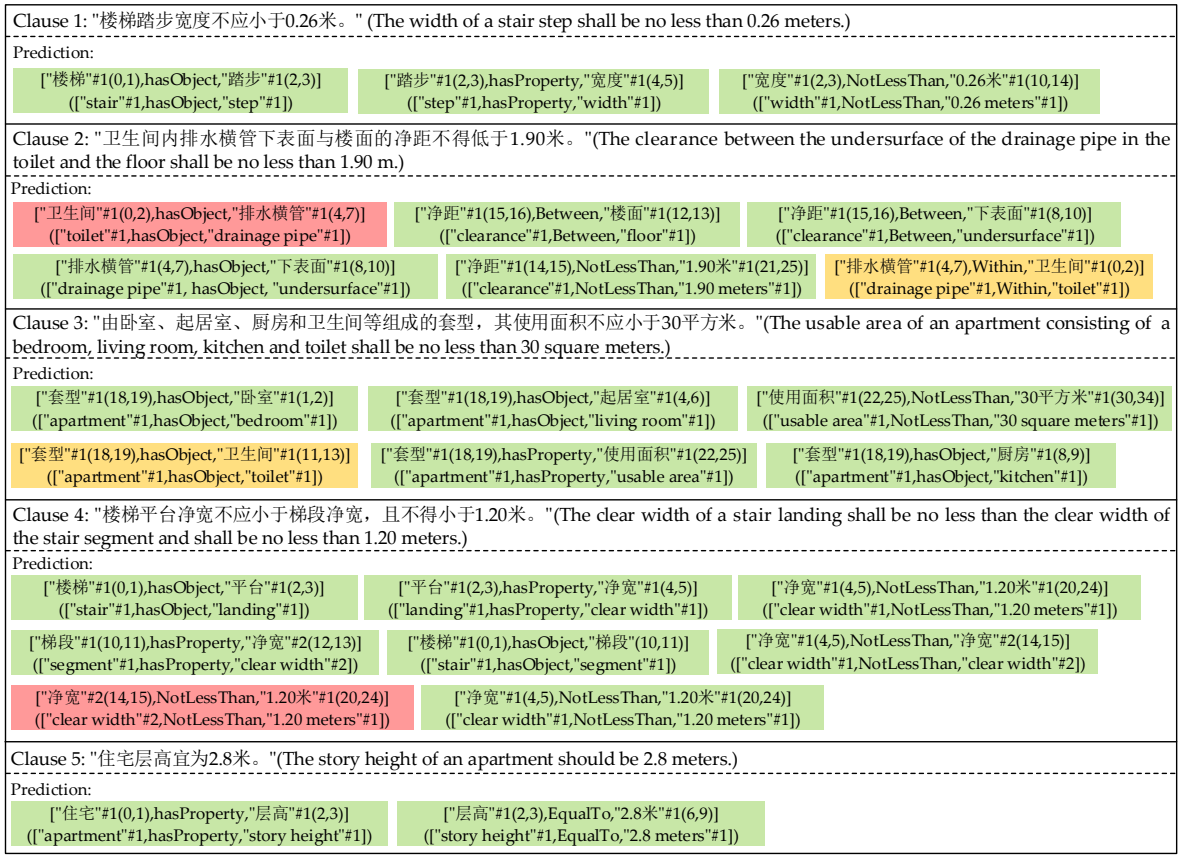


Figure 6. Results of model testing (partial).

Three metrics, that is, the recall, precision, and F-measure, were calculated to evaluate the model performance. The precision is defined as the percentage of correctly extracted triplets among the total number of extracted triplets (Equation (17)). The recall is the fraction of correctly extracted triplets among the total triplets existing in the source text (Equation (18)). The F-measure combines the recall and precision into a single measure, where β is a parameter that is used to assign relative weights to the recall and precision values (Equation (19)).

$$P = \frac{\text{number of correct triplets extracted}}{\text{total number of triplets extracted}} \tag{17}$$

$$R = \frac{\text{number of correct triplets extracted}}{\text{total number of triplets existing}} \tag{18}$$

$$F - measure = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}. \tag{19}$$

Here, the value of β was set to 1 to assign equal weights to the precision and recall. In this experiment, a triplet was considered correct only if its subject entity, object entity, and predicate relation were all correct.

The precision, recall, and F-measure scores achieved by the model in this test for each of the 14 predicate types are listed in Table 4. On average, a precision, recall, and F-measure of 0.8808, 0.8519, and 0.8661, respectively, were achieved. These results validate the effectiveness of the proposed model in extracting multiple relations and entities from building code sentences. In addition, for 7 of the 14 predicate types, the model achieved relatively high F-measure scores of more than 0.8. For the other types (e.g., "LessThan" and "GreaterThan"), the scores of less than 0.8 may be attributable to the small sample sizes for these types among all clauses.

Table 4. Precision, recall, and F-measure results obtained through model testing.

Predicate Type	In Gold Standard	Extracted	Correctly Extracted	Precision	Recall	F-Measure
hasObject	305	298	243	0.8154	0.7967	0.8060
hasProperty	406	388	357	0.9201	0.8793	0.8992
hasFeature	279	266	240	0.9023	0.8602	0.8807
AccessTo	26	19	17	0.8947	0.6538	0.7556
Within	42	44	38	0.8636	0.9048	0.8837
Outside	21	15	13	0.8667	0.6190	0.7222
Between	115	108	95	0.8796	0.8261	0.8520
AdjacentTo	30	23	21	0.9130	0.7000	0.7925
NotLessThan	278	290	266	0.9172	0.9568	0.9366
NotGreaterThan	78	77	69	0.8961	0.8846	0.8903
LessThan	26	27	18	0.6667	0.6923	0.6792
GreaterThan	27	27	19	0.7037	0.7037	0.7037
EqualTo	17	20	14	0.7001	0.8235	0.7568
MultipliedBy	24	17	16	0.9412	0.6667	0.7805
Total	1674	1619	1426	0.8808	0.8519	0.8661

Table 5 compares the test results of the proposed model with those of the five classic extraction models, from which it can be seen that the former achieved the best performance. The data in Table 5 indicate poor performance of the models proposed by Ren et al. [36] and Zheng et al. [37] since they lack the ability to cope with complex overlapping relations. In addition, Tan's model [28] can predict only a fixed number of triplets, and not all triplets can be extracted; Zeng's model [38] fails to extract entities consisting of multiple words, and the final results were significantly influenced by the word segmentation performance; and Fu's work [30] assumes that every pair of entities has an associated relation, causing the model to be misled by many nonrelated entity pairs.

Table 5. Comparison of results on the test set.

Model	Precision	Recall	F-Measure
Ren et al. [36]	0.3748	0.3202	0.3454
Zheng et al. [37]	0.5039	0.4253	0.4613
Tan et al. [28]	0.6704	0.6099	0.6387
Zeng et al. [38]	0.7098	0.6780	0.6936
Fu et al. [30]	0.7440	0.7640	0.7539
Our Model	0.8808	0.8519	0.8661

To further demonstrate the contributions of some of the components illustrated in Figure 3, one component at a time was intentionally removed to explore the impact on performance. Concretely, the word-level embedding, character-level embedding, sentence-level feature vectors, start tagging feature vectors, and joint learning framework were investigated in this way. Specifically, the start tagging feature vectors were removed when predicting the end positions, and the joint learning framework was removed by training the subject extractor and the object and predicate extractor separately, without parameter sharing.

The ablation results presented in Table 6 indicate the following: (1) The integration of the character-level and word-level embedding methods is useful for capturing the prior semantic information of each character appearing in the code clauses. (2) Introducing sentence-level feature vectors is an effective means of further encoding sentence information for use in prediction. (3) Considering the predicted start positions when predicting the end positions is beneficial, as implied by the 3.5% decline in the F-measure score observed when removing the start tagging features from the end position prediction process. (4) The joint learning framework is also effective, as implied by the 6.4% drop in the F-measure score seen when the subject extractor and the object-and-predicate extractor are trained separately.

Table 6. An ablation study on the test set.

Model	Precision	Recall	F-Measure
Whole Model	0.8808	0.8519	0.8661
-word-level embedding	0.8535	0.8005	0.8261
-character-level embedding	0.8647	0.8094	0.8362
-sentence-level features	0.8692	0.8136	0.8404
-start tagging features	0.8582	0.8065	0.8315
-joint learning	0.8287	0.7772	0.8021

4. Conclusions

Chinese code clauses frequently contain complex semantic relations in terms of the multiple relations among named entities and the overlapping relation problem, leading to difficulty for the extraction of engineering constraints. Unfortunately, existing machine-learning based methods for regulation information extraction are inadequate to address the complexity of the semantic relations between entities. In this study, a joint extraction model was proposed for extracting multiple relations and entities from code clauses.

In the proposed model, a hybrid deep learning algorithm combined with a decomposition strategy was applied. In this way, multiple relations, especially overlapping relations, can be extracted. Meanwhile, non-related pairs were excluded through the judicious recognition of the subject entities. An experiment was conducted to validate the performance of the proposed model, which achieved an average precision, recall, and F-measure of 0.8808, 0.8519, and 0.8661, respectively. The experimental results indicate that the proposed model is promising for extracting multiple relations and entities from building codes. With the proposed joint extraction model, the engineering constraints can be better extracted and then formally represented with the assistance of the ontological description of semantic relationships between named entities. In this way, the code clauses written in a natural language can be automatically processed and understood by computers.

Although a collection of domain-specific entity and relation types was investigated, only the principle types of entities and relations appearing in Chinese building codes were considered as extraction targets. In the future, more entity/relation types will be considered to test the proposed method. Although the experimental results showed a favorable outcome, the proposed methodology was only tested on a relatively small size of datasets because a significant amount of time and human efforts would be required to create a larger dataset. Thus, addressing the dataset size problem is needed in the future to further verify and refine the performance of the presented approach. In addition, this research has mainly focused on the process of information extraction; the question of how to automatically convert the extracted triplets into logical inferences for automated code compliance verification should be further explored in future work.

Author Contributions: Conceptualization, F.L. and Y.S. (Yuanbin Song); methodology, F.L. and Y.S. (Yuanbin Song); software, F.L. and Y.S. (Yuanbin Song); validation, Y.S. (Yuanbin Song) and Y.S. (Yongwei Shan); formal analysis, F.L. and Y.S. (Yuanbin Song); investigation, F.L. and Y.S. (Yuanbin Song); resources, F.L. and Y.S. (Yuanbin Song); data curation, F.L. and Y.S. (Yuanbin Song); writing–original draft preparation, F.L. and Y.S. (Yuanbin Song); writing–review and editing, Y.S. (Yuanbin Song) and Y.S. (Yongwei Shan); visualization, F.L. and Y.S. (Yuanbin Song); supervision, Y.S. (Yuanbin Song) and Y.S. (Yongwei Shan); project administration, Y.S. (Yuanbin Song); funding acquisition, Y.S. (Yuanbin Song). All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (Grant No.71271137).

Acknowledgments: The authors thank all of the reviewers for their comments on this article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Eastman, C.; Lee, J.; Jeong, Y.; Lee, J. Automatic rule-based checking of building designs. *Autom. Constr.* **2009**, *18*, 1011–1033. [CrossRef]
2. Kim, I.; Lee, Y.; Choi, J. BIM-based hazard recognition and evaluation methodology for automating construction site risk assessment. *Appl. Sci.* **2020**, *10*, 2335. [CrossRef]
3. Jiang, S.; Wu, Z.; Zhang, B.; Cha, H.S. Combined MvdXML and semantic technologies for green construction code checking. *Appl. Sci.* **2019**, *9*, 1463. [CrossRef]
4. Lee, Y.; Kim, I.; Choi, J. Development of BIM-based risk rating estimation automation and a design-for-safety review system. *Appl. Sci.* **2020**, *10*, 3902. [CrossRef]
5. Zhong, B.; Gan, C.; Luo, H.; Xing, X. Ontology-based framework for building environmental monitoring and compliance checking under BIM environment. *Build. Environ.* **2018**, *141*, 127–142. [CrossRef]
6. CORENET e-PlanCheck: Singapore's Automated Code Checking System. Available online: http://www.aecbytes.com/feature/2005/CORENETePlanCheck.html (accessed on 25 September 2020).
7. Zhong, B.; Ding, L.; Luo, H.; Zhou, Y.; Hu, Y.; Hu, H. Ontology-based semantic modeling of regulation constraint for automated construction quality compliance checking. *Autom. Constr.* **2012**, *28*, 58–70. [CrossRef]
8. Zhang, J.; El-Gohary, N.M. Semantic NLP-based information extraction from construction regulatory documents for automated compliance checking. *J. Comput. Civ. Eng.* **2016**, *30*, 04015014. [CrossRef]
9. Zhang, J.; El-Gohary, N.M. Integrating semantic NLP and logic reasoning into a unified system for fully-automated code checking. *Autom. Constr.* **2017**, *73*, 45–57. [CrossRef]
10. Zhou, P.; El-Gohary, N.M. Ontology-based automated information extraction from building energy conservation codes. *Autom. Constr.* **2017**, *74*, 103–117. [CrossRef]
11. Li, S.; Cai, H.; Kamat, V.R. Integrating natural language processing and spatial reasoning for utility compliance checking. *J. Constr. Eng. Manag.* **2016**, *142*, 04016074. [CrossRef]
12. Xu, X.; Cai, H. Semantic approach to compliance checking of underground utilities. *Autom. Constr.* **2017**, *109*, 103006. [CrossRef]
13. Zhang, R.; El-Gohary, N.M. A machine learning approach for compliance checking-specific semantic role labeling of building code sentences. In Proceedings of the 35th International Conference of CIB W78, Chicago, IL, USA, 1–3 October 2018; pp. 561–568.
14. Xu, X.; Cai, H. Semantic frame-based information extraction from utility regulatory documents to support compliance checking. In Proceedings of the 35th International Conference of CIB W78, Chicago, IL, USA, 1–3 October 2018; pp. 223–230.
15. Petruck, M.R.; Ellsworth, M. Representing spatial relations in FrameNet. In Proceedings of the First International Workshop on Spatial Language Understanding, New Orleans, LA, USA, 12–16 June 2018; pp. 41–45.
16. Papadopoulos, D.; Papadakis, N.; Litke, A. A methodology for open information extraction and representation from large scientific corpora: the CORD-19 data exploration use case. *Appl. Sci.* **2020**, *10*, 5630. [CrossRef]
17. Wang, Y.; Sun, Y.; Ma, Z.; Gao, L.; Xu, Y. An ERNIE-based joint model for Chinese named entity recognition. *Appl. Sci.* **2020**, *10*, 5711. [CrossRef]
18. Song, J.; Kim, J.; Lee, J. NLP and deep learning-based analysis of building regulations to support automated rule checking system. In Proceedings of the 35th International Symposium on Automation and Robotics in Construction, Berlin, Germany, 23–25 July 2018; pp. 586–592.
19. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. In Proceedings of the First International Conference on Learning Representations, Scottsdale, AZ, USA, 2–4 May 2013.
20. Song, J.; Lee, J.; Choi, J.; Kim, I. Deep learning-based extraction of predicate-argument structure (PAS) in building design rule sentences. *J. Comput. Des. Eng.* **2020**, *7*, 1–14.
21. Zhong, B.; Xing, X.; Luo, H.; Zhou, Q.; Li, H.; Rose, T.; Fang, W. Deep learning-based extraction of construction procedural constraints from construction regulations. *Adv. Eng. Inform.* **2020**, *43*, 101003. [CrossRef]

22. He, C.; Tan, Z.; Wang, H.; Zhang, C.; Hu, Y.; Ge, B. Open domain Chinese triples hierarchical extraction method. *Appl. Sci.* **2020**, *10*, 4819. [CrossRef]
23. Lehmann, J.; Isele, R.; Jakob, M.; Jentzsch, A.; Kontokostas, D.; Mendes, P.; Hellmann, S.; Morsey, M.; Van Kleef, P.; Auer, S.; et al. DBpedia—A large-scale, multilingual knowledge base extracted from wikipedia. *Semant. Web.* **2015**, *6*, 167–195. [CrossRef]
24. Al Qady, M.; Kandil, A. Concept relation extraction from construction documents using natural language processing. *J. Constr. Eng. Manag.* **2010**, *136*, 294–302. [CrossRef]
25. Xiong, R.; Song, Y.; Li, H.; Wang, Y. Onsite video mining for construction hazards identification with visual relationships. *Adv. Eng. Inform.* **2019**, *42*, 100966. [CrossRef]
26. International Organization for Standardization. *ISO 12006-2: 2015. Building Construction. Organization of Information about Construction Works. Part 2: Framework for Classification of Information*, 2nd ed.; International Organization for Standardization: Geneva, Switzerland, 2015.
27. Xiao, S.; Song, M. A text-generated method to joint extraction of entities and relations. *Appl. Sci.* **2019**, *9*, 3795.
28. Tan, Z.; Zhao, X.; Wang, W.; Xiao, W. Jointly extracting multiple triplets with multilayer translation constraints. In Proceedings of the AAAI Conference on Artificial Intelligence; Association for the Advancement of Artificial Intelligence (AAAI), Honolulu, HI, USA, 27 January–1 February 2019; pp. 7080–7087.
29. Dai, D.; Xiao, X.; Lyu, Y.; Dou, S.; She, Q.; Wang, H. Joint extraction of entities and overlapping relations using position-attentive sequence labeling. In Proceedings of the AAAI Conference on Artificial Intelligence; Association for the Advancement of Artificial Intelligence (AAAI), Honolulu, HI, USA, 27 January–1 February 2019; pp. 6300–6308.
30. Fu, T.; Li, P.; Ma, W. GraphRel: Modeling text as relational graphs for joint entity and relation extraction. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; pp. 1409–1418.
31. Yu, B.; Zhang, Z.; Su, J. Joint extraction of entities and relations based on a novel decomposition strategy. *arXiv* **2019**, arXiv:1909.04273.
32. MOHURD. *Standard for Classification and Coding of Building Information Model (GB/T51269-2017)*; China Architecture & Building Press: Beijing, China, 2017.
33. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
34. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Red Hook, NY, USA, 8–14 December 2017; pp. 6000–6010.
35. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
36. Ren, X.; Wu, Z.; He, W.; Qu, M.; Voss, C.R.; Ji, H.; Abdelzaher, T.F.; Han, J. Cotype: Joint extraction of typed entities and relations with knowledge bases. In Proceedings of the 26th International Conference on World Wide Web, Perth, Australia, 3–7 May 2017; pp. 1015–1024.
37. Zheng, S.; Wang, F.; Bao, H.; Hao, Y.; Zhou, P.; Xu, B.; Barzilay, R.; Kan, M.Y. Joint extraction of entities and relations based on a novel tagging scheme. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Vancouver, BC, Canada, 30 July–4 August 2017.
38. Zeng, X.; Zeng, D.; He, S.; Liu, K.; Zhao, J. Extracting relational facts by an end-to-end neural model with copy mechanism. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Melbourne, Australia, 15–20 July 2018.