A deep neural network-based method for deep information extraction using transfer learning strategies to support automated compliance checking

Ruichuan Zhang[a]; and Nora El-Gohary[b]

[a] Graduate Student, Department of Civil and Environmental Engineering, University of Illinois at Urbana-Champaign, 205 N. Mathews Ave., Urbana, IL 61801, United States. E-mail: rzhang65@illinois.edu.

[b] Associate Professor, Department of Civil and Environmental Engineering, University of Illinois at Urbana-Champaign, 205 N. Mathews Ave., Urbana, IL 61801, United States (corresponding author). E-mail: gohary@illinois.edu; Tel: +1-217-333-6620; Fax: +1-217- 265-8039.

Abstract

Existing automated compliance checking (ACC) systems require the extraction of requirements from regulatory documents into computer-processable representations. These information extraction (IE) processes are either fully manual, semi-automated, or automated. Semi-automated and manual approaches typically use manual annotations or predefined IE rules, which lack sufficient flexibility and scalability; the annotations and rules typically need adaptation if the characteristics of the regulatory document change. There is, thus, a need for a fully automated IE approach that can achieve high and consistent performance across different types of regulatory documents for supporting ACC. To address this need, this paper proposes a deep neural network-based method for deep information extraction – extracting semantic and syntactic information elements – from regulatory documents in the architectural, engineering, and construction (AEC) domain. The proposed method was evaluated in extracting information from multiple regulatory documents in the AEC domain. It achieved average precision and recall of 93.1% and 92.9%, respectively.

Keywords: Code checking; Information extraction; Deep learning; Transfer learning.

1 Introduction

Existing automated code checking (ACC) systems have achieved different levels of accuracy, automation, and coverage. However, they all require the extraction of requirements from regulatory documents, such as building codes, energy conservation codes, and specifications, into computer-processable representations. The information extraction (IE) processes in many of the existing ACC systems are still fully manual. For example, Solibri Model Checker requires users to read the building code and then manually convert the building-code requirements into computer-processable representations by filling in predefined templates. The IE processes in other ACC systems are semi-automated or automated. However, these processes still rely on manual annotations or manually defined IE rules. For example, SmartCode requires that code-checking professionals annotate regulatory information using the requirement, application, selection, and exception (RASE) markups [1] manually, and the annotated building-code text is then converted into a computer-processable form using predefined rules. The state-of-the-art rule-based ACC systems by Zhang and El-Gohary [2] and Zhou and El-Gohary [3] use IE rules developed by experts based on the syntactic information of building-code sentences (e.g., part-of-speech tags) and construction-domain ontologies to extract a defined set of semantic information elements. Despite the high IE performance they have achieved, the annotation-based or rule-based approaches, by nature, lack sufficient flexibility and scalability; the annotations and rules typically need adaptation if the characteristics of the building-code text change.

Machine learning-based IE methods, instead of relying on manual annotation or hand-crafted rules, use machine learning models to automatically capture the underlying syntactic and semantic patterns of the text. A machine learning-based method is, thus, expected to be more flexible and scalable compared to the annotation-based or rule-based IE methods – saving the initial effort to develop the annotations or rules, as well as the maintenance effort that would be required to adapt the annotations or rules across different types of regulatory documents or extraction tasks. However, not any machine learning algorithm would be suitable for this information extraction task. Using machine learning to extracting regulatory information from building codes to support ACC is challenging from two perspectives. First, existing machine learning-based IE methods in the AEC domain are only able to support shallow IE, where partial information (e.g., bridge deficiency-related entities [4]) is extracted from the text. However, ACC systems require deep IE, where the entire meaning of the text is captured for complete and correct extraction of the requirements [2]. Defining all semantic entities (e.g., subject, compliance checking attribute, reference) that can capture the full meaning of all types of requirements, and extracting them using machine learning methods, helps achieve such level of complete extraction. Second, building codes have hierarchically complex syntactic and semantic structures. Compared to general domain text, building-code sentences typically have deeply nested syntactic and semantic structures, including recursive clauses, conjunctive and alternative obligations, and multiple exceptions [3]. Recent efforts (e.g, [5]) have shown that deep neural networks are capable of learning the complex syntactics and semantics of the natural language. Thus, there is a need to explore the use of deep neural networks in deep IE for supporting ACC.

To address this need, this paper proposes a deep neural network-based method for fully automated extraction of semantic and syntactic information elements from regulatory documents for supporting ACC in the architecture, engineering, and construction (AEC) domain. The deep learning models, which have significantly more parameters compared to traditional machine learning models, typically need a larger scale of data for training. However, there are no such annotated training datasets in the AEC domain, and creating these datasets would be highly expensive. To solve this problem, the proposed method uses transfer learning strategies to enable the training of deep neural network models on both domain-general and AEC-specific annotated data. On one hand, domain-general data (i.e., the source-domain data in the context of transfer learning) are large in scale and rich in syntactic and semantic patterns, which helps train the models to deal with various text patterns across different regulatory documents for increased IE performance, flexibility, and scalability. However, the domain-general data are relatively different from the AEC-specific data in terms of vocabularies, syntactics, and semantics. On the other hand, AEC-domain data (i.e., the target-domain data in the context of transfer learning) are the target data to be analyzed, but they are much smaller in scale and lack syntactic and semantic richness, which would limit the flexibility and scalability of the models if they are solely used for training. The proposed approach, thus, takes the best of both worlds.

The proposed deep neural network-based IE approach consists of four main steps: (1) prepare training data from both outside of the AEC domain (i.e., the source-domain data) and within the AEC domain (i.e., the target-domain data) and testing data; (2) develop a base deep IE model – a deep neural network model that consists of long short term memory networks (LSTM) and conditional random fields (CRF) for automatically extracting semantic and syntactic information elements from regulatory documents; (3) train the deep IE model using different transfer learning strategies including feature-based and model-based ones; and (4) evaluate the deep IE performance using precision, recall, and F1 measure.

2 Background

2.1 Information extraction

Information extraction (IE) aims to automatically extract structured information (e.g., entities and attributes that describe the entities) from text data, which are often unstructured and thus are not processable and understandable by computers [6]. Existing IE methods can be classified into two groups: rule-based and machine learning-based methods. Rule-based IE approaches rely on pattern-matching rules that are developed based on semantic and syntactic knowledge. The IE rules are often manually designed. For example, Fader et al. [7] developed IE rules based on the syntactic and lexical features of the text to extract assertions from the Web for supporting commonsense knowledge and question answering. The ClausIE by Del Corro and Gemulla [8] consisted of IE rules built upon English grammar and dependency parsing of sentences to extract arguments from text. The IE system by Gutierrez et al. [9] integrated IE rules and error detection rules built upon biology-related ontologies to extract facts from text in the biological domain. A few other research efforts explored a number of techniques to reduce the cost of creating IE rules, such as learning IE rules from plain text using statistical learning algorithms [10], designing simple programming languages and interactive environments for rules [11], and integrating existing rule programming languages and natural language processing applications in one rule development platform [12].

Machine learning-based methods, rather than relying on IE rules, employ machine learning models to automatically learn the syntactic and semantic patterns from training text data – and the trained IE models are then used to extract the target information from new, unseen text data. The most commonly used machine learning-based methods formulate the IE problem as a sequence labeling problem, where each word in a sentence is assigned a label using supervised learning algorithms. Examples of IE approaches using traditional supervised learning algorithms, together with handcrafted syntactic and semantic features, include a hidden Markov algorithm-based named entity recognition (NER) method by Zhou and Su [13], a support vector machine-based NER method by Li et al. [14], and a CRF-based IE method by Finkel et al. [15].

2.2 Deep learning in text analytics

Deep learning methods use computational models that consist of multiple layers to capture different levels of information representations from large-scale data [16]. Deep learning methods have drastically improved the state-of-the-art performance in automatically processing and understanding different types of data, including image and text, and meanwhile reduced or eliminated the manual effort in feature engineering compared to traditional machine learning methods. Recurrent neural networks (RNN) and variants such as gated recurrent units (GRUs) [17] and LSTM [18] are deep neural networks that use internal states to process sequences of input data. They have been widely used in text analytics tasks including semantic and syntactic analysis (e.g., bidirectional LSTM and multilayer perceptron for dependency parsing and part-of-speech (POS) tagging [19]), and partial or shallow IE (e.g., bidirectional LSTM and CRF for extracting named entities [20]). Examples of RNN-based IE efforts include a domain-specific event detection method using convolutional neural networks [21], an NER method using LSTM and CRF [20], and an entity and relation extraction system using bidirectional LSTM [22].

Most recently, the Transformer [23] and transformer-based models and methods have been proposed, which allow training language models on large-scale text data much faster by abandoning complex RNN and solely relying on the attention mechanisms. For example, the OpenAI's generative pre-trained transformer (GPT) [24] and Google's bidirectional encoder representations from transformers (BERT) [25], as well as many of their variants (e.g., XLNet [26], DistilBERT [27], ALBERT [28]), which improve on either the performance or the training speed, have achieved the state-of-the-art performance in various text analytics tasks (e.g., machine translation [29], question answering [30], and information retrieval [31]).

A limited number of research efforts have been focused on deep learning-based methods to solve text analysis problems in the AEC domain. For example, Zhang and El-Gohary [32] used an RNN-based approach to extract requirement hierarchies from building-code sentences for supporting compliance checking. Pan and Zhang [33] developed RNN-based models to mine information from building information modeling (BIM) log data to support design decision making. Bang and Kim [34] developed models that consist of convolutional neural network (CNN) and LSTM layers to automatically generate time-spatial and visual context-based descriptions given construction site images for supporting construction site management.

2.3 Transfer learning

One challenge for deep learning-based IE is that the models typically need large annotated text data, which require significant time and effort to prepare. Such annotated data are scarce in many domains, including the AEC domain, which hinders the use of deep learning for domain-specific IE. Existing annotated datasets have mostly been developed for general natural language processing (NLP) applications (e.g., the Penn Treebank datasets for multiple syntactic and semantic analysis tasks [35], the CoNLL-2003 dataset for language-independent NER [36], and the CoNLL-2005 for semantic role labeling [37]), which are not sufficient for many domain-specific applications such as IE for ACC. To address this problem, various research efforts have been undertaken to leverage labeled data from other domains using transfer learning strategies.

Transfer learning aims to transfer knowledge for solving certain domain-specific tasks by leveraging existing labeled data of some related tasks or domains [38]. Transfer learning enables the training of machine learning models using large-scale, pattern-rich, and annotated training data that are from source domains that are different from the target domain (e.g., the AEC domain). Thus, transfer learning improves both the performance and the flexibility and scalability of the machine learning models, as well as reduces the cost of preparing annotated training data for the target domain. Transfer learning strategies can be classified into three types based on how the knowledge is transferred from the source domains to the target domain: instance-based, feature-based, and model-based strategies.

Instance-based strategies reweight or resample the source-domain data to be similar to the target-domain data (e.g., the boosting method for cross-domain text classification [39]), which are then used for training the machine learning models. Feature or representation-based strategies discover transferable features or representations that are discriminative for both the source and the target domains through a new machine learning model (e.g., the global vectors for word representation model [40] and the deep contextualized word representations [41]). Model-based strategies reapply the partial deep neural networks – those layers trained on the source-domain data – in the target domain by adapting the models using target-domain data. Examples of methods for model adaptation include finetuning the pretrained CNN-based image classification models (e.g., [42-43]), finetuning the pretrained Transformer-based models (e.g., GPT, BERT, or their variants) for specific downstream text analytics tasks (e.g., [29-31]), and training the sequence labeling model on source-domain and target-domain data alternatingly (e.g., [44]).

Transfer learning strategies have been used to solve computer vision and NLP problems such as sequence labeling (e.g., [44]), text classification (e.g., [39]), and sequence-to-sequence learning (e.g., [29-30]). In the AEC domain, transfer learning strategies have been mainly used to solve computer vision problems (e.g., [42-43]).

3 State of the art and knowledge gaps in information extraction in the construction domain

Rule-based methods have been developed for solving various IE problems in the AEC domain. For example, Al Qady and Kandil [45] developed rules, which use syntactic features, for shallow parsing to extract concept relations from construction contract documents for improving electronic document management such as document categorization and retrieval. Zhang and El-Gohary [2] and Zhou and El-Gohary [3] developed IE rules, which use semantic and syntactic features, to extract semantic information elements from regulatory documents such as building codes, energy conservation codes, and specifications for supporting ACC. Lee et al. [46] developed rules, which use syntactic parsing and predefined lexicon features, to extract poisonous clauses from construction contracts for supporting contract management. Despite the state-of-the-art performance levels many of them have achieved (e.g., nearly 100% recall reported by Zhang and El-Gohary [2] and Zhou and El-Gohary [3]), the rule-based approaches are difficult to scale to a variety of documents due to the relatively limited and inflexible patterns that are used to develop the rules. In general, when the type of regulatory document or the characteristics of the text change, although some of the IE rules could be reused, most of these rules will require significant retesting and possibly modification or addition. The lack of sufficient flexibility and scalability becomes a potential obstacle for using ACC systems built on rule-based IE, especially given the fact that building codes are updated frequently and vary across different regions.

Recently, a limited number of machine learning-based methods have been developed for solving IE problems in the AEC domain. For example, Liu and El-Gohary [4] developed a semi-supervised machine learning-based method to extract entity information from bridge inspection reports for supporting bridge deterioration prediction. Zhang and El-Gohary [47] developed a supervised learning-based method to extract semantic roles including entities and relations from regulatory documents for supporting ACC. Kim and Shi [48] developed a supervised learning-based method to extract knowledge from construction accident cases. Despite the importance of these efforts, there are three knowledge gaps that this paper aims to address. First, the aforementioned methods can be classified as shallow because they only extract partial information from the text, and thus they cannot be directly used for capturing the entire meaning of the text, which is essential for IE for ACC. Second, they use traditional machine learning algorithms such as CRF, which has been outperformed by deep neural networks such as RNN in many text analytics tasks including partial or shallow IE. Thus, there is a need to explore the use of deep neural networks in deep IE for supporting ACC. Third, there is generally a lack of labeled training data in the AEC domain, which is especially a challenge for deep neural networks because they require larger training datasets than those required for traditional algorithms. Thus, there is a need for techniques to leverage the larger-size and pattern-rich data that exist in other domains to help address this challenge while reducing the human-labeling effort.

4 Proposed semantic and syntactic information elements for deep information extraction for supporting ACC

In this study, two types of information elements, semantic and syntactic information elements, are used to represent the building-code requirements. The semantic information elements define the building-code requirements that are described in the natural language building-code sentences. In this study, a subset of the semantic information elements proposed by Zhang and El-Gohary [2] were utilized, including six of the essential semantic information elements (as shown in Table 1): subject, compliance checking attribute, deontic operator indicator, comparative relation, quantity value, and quantity unit. Two new semantic information elements were added: subject relation and reference. Subject relation extends the original quantity relation to relations that apply to both quantitative and nonquantitative requirements. Reference extends the scope of existing ACC efforts to cover cross references that commonly exist in requirements. The secondary semantic information elements such as subject restrictions and quantity restrictions [2] were not utilized, because compared to the study by Zhang and El-Gohary, this study further granularizes the regulatory information represented by the secondary semantic information elements using the proposed information elements, and thus there is no need to include secondary elements. The syntactic information elements are used in the English sentence to form grammatically correct building-code sentences but do not directly contribute to defining the meaning of the building-code requirement. The syntactic information elements include three types of logic operator indicators – conjunctions (e.g., "and"), disjunctions (e.g., "or"), and negations (e.g., "not") – and syntactic units such as some of the pronouns (e.g., "the"), adverbs (e.g., "so"), prepositions (e.g., "of"), and conjunctions that introduce a clause (e.g., "that"). These syntactic information elements better capture the syntactic structures of requirements (especially the deeply nested ones), which helps better understand the full meaning of the requirements. Fig. 1 shows example sentences from the International Building Code (IBC), International Energy Conservation Code (IECC), and Americans with Disabilities Act (ADA) Standards, and how the sentences are annotated using the proposed semantic and syntactic information elements.

Table 1. Semantic Information Elements for Representing Requirements for Compliance Checking Purposes [2]

Semantic information element	Definition

Subject	An ontology concept representing a thing (e.g., building element) that is subject to a particular requirement
Compliance checking attribute	An ontology concept representing a specific characteristic of a "subject" that is checked for compliance
Deontic operator indicator	A term or phrase that indicates the deontic type of the requirement (i.e., obligation, permission, or prohibition)
Comparative relation	A term or phrase for comparing quantitative values, including "greater than or equal to," "greater than," "less than or equal to," "less than," and "equal to"
Quantity value	A numerical value that defines the quantity
Quantity unit	The unit of measure for a "quantity value"
Subject relation	A term or phrase that defines the type of relation between two subjects, a subject and an attribute, or a subject or an attribute and a quantity
Reference	A term or phrase that denotes the mention or reference to a chapter, section, document, table, or equation in a building-code sentence (e.g., "Section 1312" in "the revolving door shall comply with Section 1312")

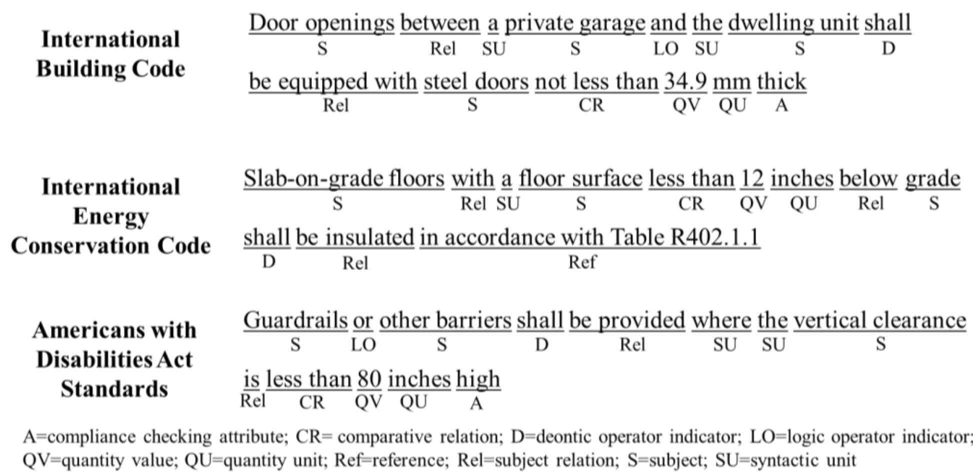

Fig. 1. Example building-code sentences annotated with the proposed syntactic and semantic information elements.

5 Proposed deep neural network-based method for deep IE from regulatory documents

The proposed deep learning-based method for deep IE from regulatory documents consists of four primary steps, as illustrated in Fig. 2: data preparation, base deep IE model development, model adaptation and training using transfer learning strategies, and deep IE performance evaluation.

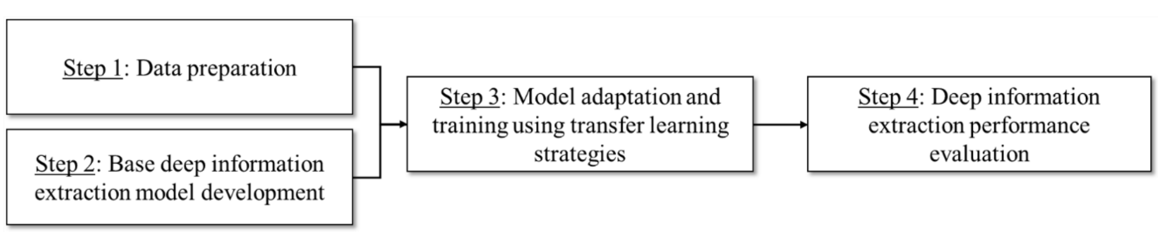

Fig. 2. Proposed deep neural network-based method for deep information extraction from regulatory documents.

5.1 Data preparation

5.1.1 Target-domain data preparation

The target-domain data – building-code sentences that are annotated with the proposed semantic and syntactic information elements – were prepared for both training and testing the IE models. The data were prepared following four steps: corpus development, data preprocessing, sentence selection, and sentence annotation. First, a small building-code corpus was developed, which consists of sentences from multiple regulatory documents, including the IBC, IECC, ADA Standards, and IBC amendments (e.g., Champaign building code amendments). To construct the corpus, all documents were converted to the text file format (i.e., .txt) and combined into a single file. Second, the following four preprocessing techniques were used: data cleaning, sentence segmentation, sentence tokenization, and sentence filtering. Data cleaning aims to remove the noises created due to the conversion of the non-textual parts (e.g., figures) of the regulatory documents. Sentence segmentation aims to detect the sentence boundaries (e.g., punctuations) and segment the text into sentences. Sentence tokenization aims to further split the sentences into tokens (e.g., words). Sentence filtering aims to remove the sentence or sentence fragments that are not requirements (e.g., headings). The Natural Language Toolkit (NLTK) in Python was used for sentence segmentation and tokenization. Third, a group of building-code sentences, which consists of about 15,000 words, were randomly selected from the developed corpus. The selected sentences have different levels of computability. Computability is defined as the ability of the building-code sentence to be represented and processed by a computer in an effective manner [49]. Fourth, a group of four participants with both domain knowledge (especially codes and regulations) and NLP knowledge – the first author and three experts including two from academia (faculty) and one from industry – manually annotated the selected sentences with the proposed semantic and syntactic information elements. The beginning-inside (BI) labeling scheme was adopted, where "B" indicates that the word is at the beginning of an information element, and "I" indicates that the word is inside of an information element. For example, the "door openings", which is a subject, is annotated as "B-Subject I-Subject", meaning that the word "door" is the beginning of a subject and the word "openings" is inside of a subject. The inter-annotator agreement was 80% in F1 measure, which indicates the reliability of the annotations [50]. The discrepancies among the annotations were then discussed and resolved to reach consensus on the final annotations. After annotation, the target-domain data was split into two sets using a 9:1 ratio: training and validation dataset and testing dataset. A ten-fold cross validation was performed, further splitting the first dataset into a training set (for training the model) and a validation set (for tuning the hyperparameters of the model). The testing dataset was used for evaluation.

5.1.2 Source-domain data preparation

The source-domain data, English sentences that are *not* from the AEC domain and are already annotated with different labels or markups (i.e., other than the proposed syntactic and semantic information elements), were prepared for training the IE model. The Penn Treebank [35] were used, which consist of over 100,000 English sentences that were collected from the Wall Street Journal and are annotated with POS tags. The Penn Treebank data are suitable for training the IE models for two reasons. First, the POS-tag annotations indicate the syntactic roles that words play in a sentence, which can be used for the syntactic and semantic analysis of the text. Second, compared to the target-domain data, the Penn Treebank data are large in scale and rich in syntactic and semantic patterns. The entire source-domain data were used for training the IE models using transfer learning strategies.

5.2 Base deep information extraction model development

The deep neural network model – bidirectional LSTM with CRF [51] – was selected and adapted as the base IE model. The base model, thus, consists of three main components: the input layer, encoding layer, and output layer, as depicted in Fig. 3. The selections of the layers were conducted based on the scales and syntactic and semantic characteristics of the specific source and target data used in the training of the model, as discussed in the following subsections.

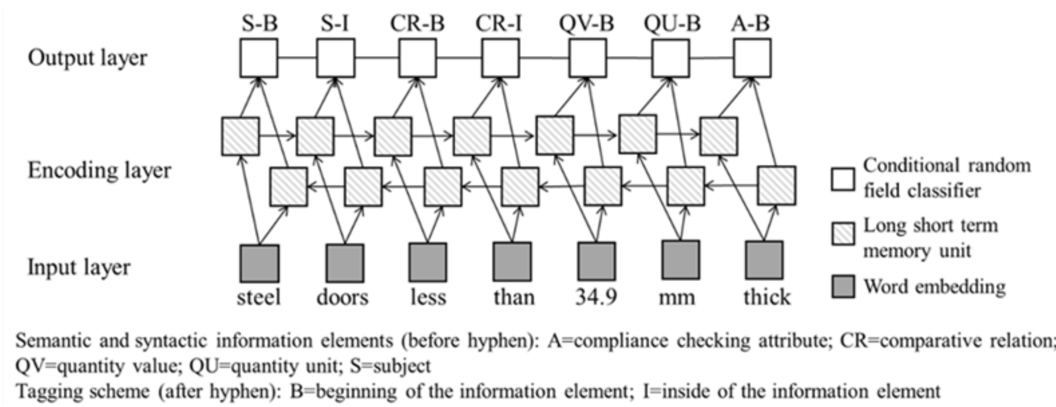

Fig. 3. The architecture of the base deep information extraction model.

5.2.1 <u>Input layer</u>

The input layer aims to represent the semantics of each word in a vector representation for deep neural network computation purposes. To better capture the semantic information of the words in the target-domain training data, which are of relatively small scale, a word-embedding layer and a character-embedding layer were added to the input layer. The word-embedding layer aims to learn the vector representation of each token (e.g., word or punctuation). The character-embedding layer aims to first learn the vector representation of each letter, digit, or symbol in the training data, and then feed the vector representations of all letters, digits, and symbols contained in a token into an LSTM layer to generate a second vector representation to represent this token.

For each token, the final output of the input layer is a vector representation formed by concatenating the vector representation generated by the word-embedding layer and the vector representation generated by the character-embedding layer.

5.2.2 Encoding layer

The encoding layer aims to further learn the contextual vector representation of each word that is discriminative in terms of the IE task, using the vector representations of both the current word and the context words generated by the input layer. To better capture the semantic information of the words in the target-domain training data, which are of relatively small scale, two LSTM layers were added to the encoding layer. To improve the ability of the IE model to deal with long-term syntactic and semantic dependencies that exist in hierarchically complex building-code sentences, the vector representations of both forward and backward context words were used when encoding the contextual vector representation of the current word via the bidirectional LSTM architecture – where one LSTM layer is forward and the other layer is backward. For each input building-code sentence, the representations encoded by the forward LSTM layer are a sequence of vectors $[f_1, f_2, \ldots, f_T]$, and the representations encoded by the backward LSTM layer are another sequence of vectors $[b_1, b_2, \ldots, b_T]$, based on which the representations generated by the encoding layer are $[h_1, h_2, \ldots, h_T]$, where h_t is the direct sum of f_t and b_t [20] and T is the size of the LSTM layers.

To improve the model's ability to reduce overfitting, a recurrent dropout layer was added to the encoding layer. The recurrent dropout layer drops a random fraction of the LSTM units in the encoding layer during the training of the IE model, according to a dropout probability d. Typically, the dropout probability is set to be smaller than 0.5, which means that less than half of the LSTM units are dropped and the rest of the LSTM units are retained. The recurrent dropout layer is disabled during the testing and future use of the IE model (i.e., use in the ACC system), which means all the LSTM units in the encoding layer are used for generating the contextual vector representations of the tokens in the building-code sentences.

5.2.3 Output layer

The output layer aims to predict the type of syntactic and semantic information elements using the BI labeling scheme for each token in the building-code sentence, given the contextual vector representations of the tokens in the sentence generated by the encoding layer. To better capture the semantic and syntactic dependencies that exist in hierarchical complex building-code sentences, a CRF layer was added to the output layer. The cross-entropy loss was chosen as the objective function and was minimized during the training of the IE model. The cross-entropy loss L describes the difference between the labels (i.e., the type of semantic information elements using the BI labeling scheme or the POS tags) in the training data, denoted as y, and the labels predicted by the model θ, denoted as c, based on the input building-code sentence x, as shown in Eq. (1), where D is a batch of the training data, C is the set of all the possible labels, and $p_\theta(c|x_i)$ is the conditional probability of c given the input sentence x generated by the CRF layer in the IE model with parameters θ, and $1_{y=c}$ is the indicator function, which returns 1 when y and c are equal, and returns 0 when y and c are not equal.

$$L(\theta) = \frac{1}{|D|} \sum_{x,y \in D} \sum_{c \in C} 1_{y=c} \log p_\theta(c|x_i) \tag{1}$$

Given a building-code sentence and a trained IE model, the corresponding sequence of labels was predicted by searching the optimal sequence of labels that maximizes the sum of the conditional log probabilities $\log p_\theta(c|x_i)$ computed by the CRF layer.

5.3 Model training using transfer learning strategies

To enable the training of the base IE model on both the source-domain and the target-domain training data, the model was further adapted and trained using different transfer learning strategies. Based on the structure of the base IE model, four transfer learning strategies, belonging to two types – feature-based and model-based strategies – were selected for testing, as summarized in Table 2.

Table 2. Transfer Learning Strategies Adopted for Training the Base Deep Information Extraction Model

Transfer learning strategy	Type of strategy	Modification of the base deep information extraction model
Fixed pretrained word embeddings	Feature-based	Initially replace the word-embedding layer with pretrained word embeddings; fix the word-embedding layer
Trainable pretrained word embeddings	Feature-based	Initially replace the wording-embedding layer with pretrained word embeddings
Two-stage training	Model-based	Replace the conditional random field (CRF) layer used in the first stage of the training with a new layer
Alternating training	Model-based	Attach two separate CRF layers to the encoding layer

5.3.1 Feature-based transfer learning strategy

Feature-based transfer learning strategies were selected to directly transfer the semantic information contained in the source-domain data to the target-domain data in the word-embedding layer of the base IE model. Pretrained word embeddings are vector representations of words learned on a large, cross-domain corpus by training a machine learning model on the corpus. The most commonly used machine learning model to generate pretrained word embeddings is the Global Vectors for Word Representation (GloVe) algorithm [40], where the training is performed on aggregated global word-word co-occurrence statistics from a large cross-domain corpus, and the resulting representations capture the contextual information of the words in the corpus. The word embeddings that were learned by applying the GloVe algorithm on a corpus consisting of Wikipedia 2014 and Gigaword 5 were adopted. The adopted word representations consist of vector representations of 40,000 uncased English words, which have a dimension of 50.

Two feature-based transfer learning strategies were adopted for training the deep IE model: the fixed pretrained word-embedding strategy and the trainable pretrained word-embedding strategy. The fixed pretrained word-embedding strategy aims to keep the weights in the input layer corresponding to the pretrained word embeddings not updated during the training of the deep neural networks. On the other hand, the trainable pretrained word-embedding strategy aims to use the pretrained word embeddings to initialize the weights in the input layer and then update the weights during the training. The performance of the two strategies depends on the relativeness of the corpus that is used to learn the pretrained word embeddings to the domain-specific text and the complexity of the syntactics and semantics in the domain-specific task.

5.3.2 Model-based transfer learning strategy

Model-based transfer learning strategies were selected to indirectly transfer the semantic information contained in the source-domain data to the target-domain data in the input layer and embedding layer of the base IE model. Two model-based transfer learning strategies were adopted for training the IE model: a two-stage training strategy and an alternating training strategy. In the two-stage training strategy (as illustrated in Fig. 4), the IE model was trained in two separate stages. In the first stage, the model was trained on the source-domain data. The first-stage training was stopped if the difference between the training losses of two consecutive training epochs is smaller than the threshold (i.e., 0.01), or the training reaches 50 epochs, where an epoch is defined as training the model on the entire source-domain data. In the second stage, the output layer of the trained model (i.e., source output layer) was replaced by a new output layer (i.e., target output layer), and the model was trained on the target-domain data. In the second stage, only the output layer was trainable, and the other two layers (i.e., the input layer and the encoding layer) were not – i.e., the parameters of these two layers were not updated during the training. The second-stage training was stopped if the difference between the training losses of two consecutive training epochs is smaller than the threshold (i.e., 0.01), or the training reaches 50 epochs, where an epoch is defined as training the model on the entire target-domain data.

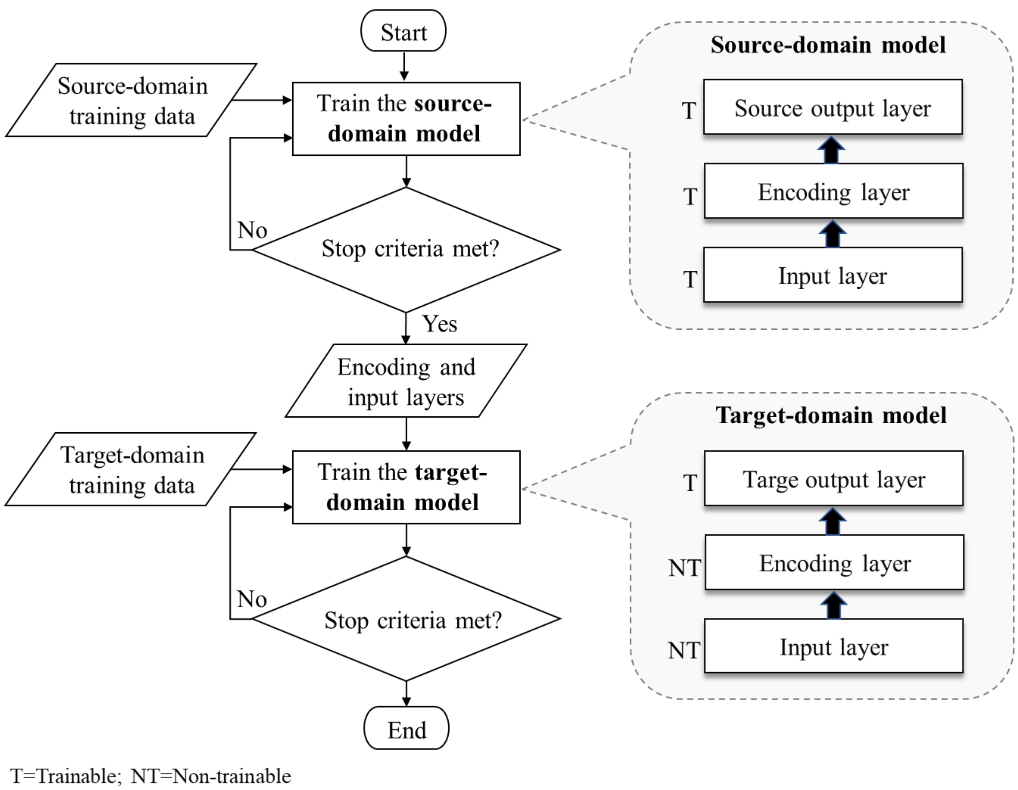

Fig. 4. Two-stage training strategy and model requirements.

In the alternating training strategy (as illustrated in Fig. 5), the IE model was trained on the source-domain and the target-domain training data in an alternating manner. The model had two separate output layers – one layer is used when the model is trained on the source-domain data (i.e., source output layer) and the other layer is used when the model is trained on the target-domain data (i.e., target output layer). In each training iteration, there is an alternating probability p that the model is trained on a selected batch of source-domain data, and a probability of $(1-p)$ that it was trained on a selected batch of target-domain data, where the total number of iterations is equal to the size of the training data divided by the size of a batch of training data. Typically, the alternating probability p is close to 1, meaning the model is more frequently trained on source-domain data rather than target-domain data, to capture as much syntactic and semantic patterns from the relatively large-scale source-domain data, and to prevent overfitting on the relatively small-scale target-domain data. The training was stopped if the difference between the training losses of two consecutive epochs when the model is trained on the target-domain data is smaller than the threshold (i.e., 0.01), or the training on the target-domain data reaches 50 epochs, where an epoch is defined as training the model on the entire target-domain training data.

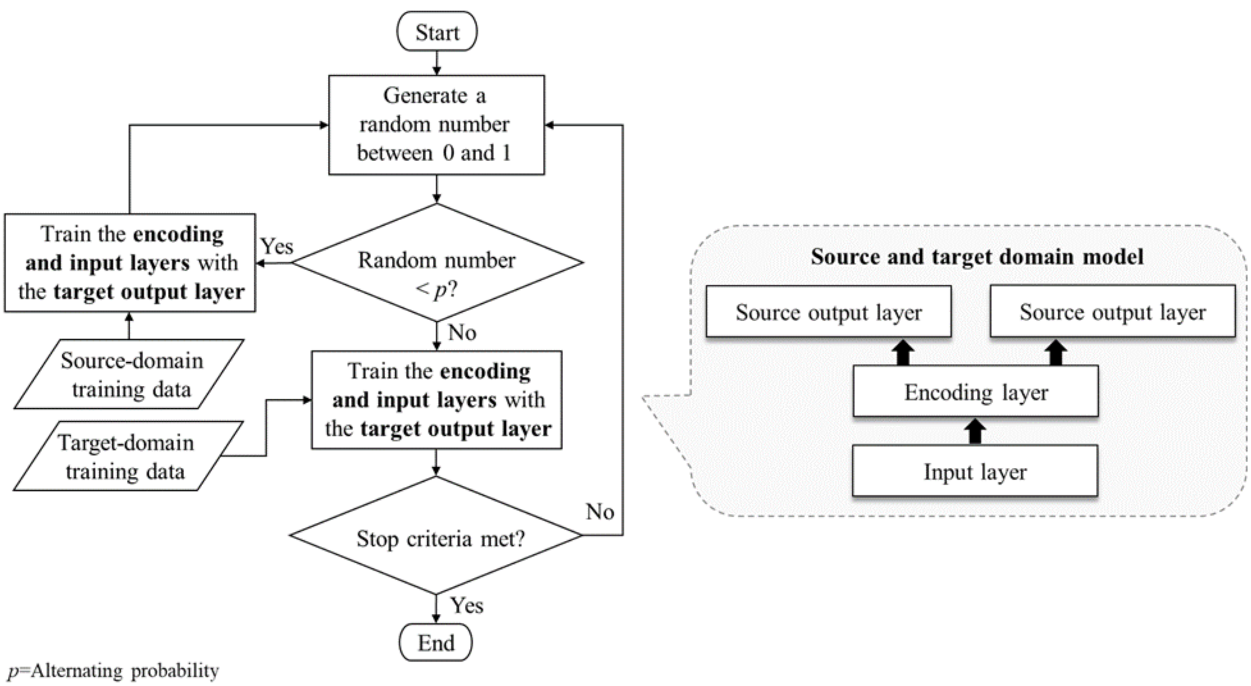

Fig. 5. Alternating training strategy and model requirements.

5.4 Deep information extraction and evaluation

To test and evaluate the proposed model, the information was extracted following two simple steps (Fig. 6). First, the building code was preprocessed into sentences, where each preprocessed sentence consisted of a sequence of tokens (e.g., words, numbers, punctuation marks). Second, the trained deep IE model automatically extracted the semantic and syntactic elements in the sentences.

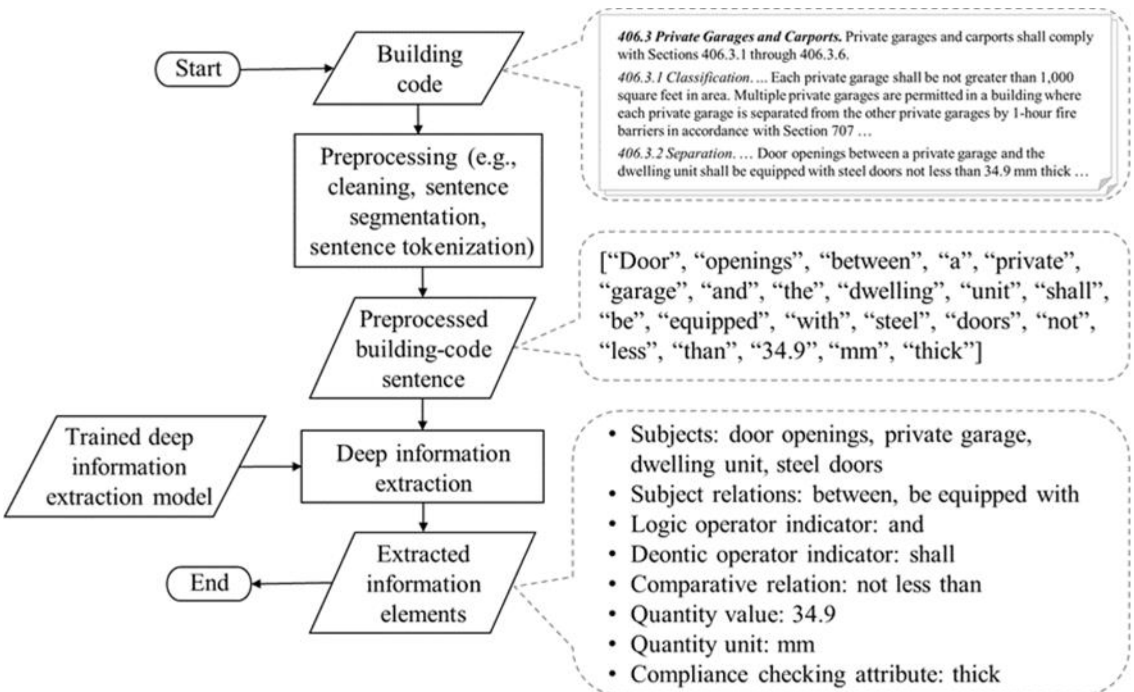

Fig. 6. Deep information extraction using the proposed method.

Three metrics were used to evaluate the IE performance: precision, recall, and F1 measure, as shown in Eq. (2) to (4), where for a specific type of syntactic and semantic information element E, TP is the number of true positives (i.e., number of words correctly labeled as E), FP is the number of false positives (i.e., number of words incorrectly labeled as E), and FN is the number of false negatives (i.e., number of words not labeled as E but should have been) [52].

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (4)$$

6 Experimental results

6.1 Deep information extraction model hyperparameter optimization

The deep IE models and transfer learning strategies were implemented using Keras built in Python 3 and run using the Tesla K80 GPU provided in the Google Colaboratory. A ten-fold cross validation was conducted for optimizing the model hyperparameters. The optimized main hyperparameters for the deep IE models are shown in Table 3.

Table 3. Optimized Main Hyperparameters for the Deep Information Extraction Models

Hyperparameter	Value
Batch size for the source-domain training data	30
Batch size for the target-domain training data	30
Size of the word-embedding vector representation	50
Size of the character-embedding vector representation	20
Size of the long short term memory layer in the encoder layer	50
Type of activation functions	rectified linear unit (ReLU)
Maximum length of input sentences	75
Maximum length of input words	20
Recurrent dropout rate	0.1
Alternating probability when training the deep information extraction models using alternating training strategy	90%
Training loss difference threshold	0.01

6.2 Comparison of the performances of the proposed method with different transfer learning strategies

To determine the optimal transfer learning strategies for the proposed deep IE method, six different combinations of strategies were implemented and tested for comparative evaluation, as shown in Table 4: two-stage training with no feature-based strategy (SC1), alternating training with no feature-based strategy (SC2), two-stage training with trainable pretrained word embeddings (SC3), alternating training with trainable pretrained word embeddings (SC4), two-stage training with fixed pretrained word embeddings (SC5), alternating training with fixed pretrained word embeddings (SC6). During the training of the model, the hyperparameters were set as per Table 3. The proposed deep IE method achieved the highest performance when the strategy combination SC4 was adopted. The results indicate that, first, the differences between the semantic and syntactic characteristics of the source-domain and target-domain data have a significant impact on the choice of transfer learning strategies. Second, the two-stage training strategy might cause the IE model to overfit to the source-domain data and underfit to the target-domain data. Third, the pretrained word embeddings contribute to the model's ability to capture the semantic and syntactic patterns in both the source-domain and target-domain data; however, they are still not able to bridge the gap between the two domains (i.e., the general domain and the AEC domain).

According to the aforementioned results, the proposed IE method uses the optimized hyperparameters in Section 6.1 (e.g., recurrent dropout rate as 0.1, alternating probability as 90%) and the transfer learning strategy combination SC4. For the remaining experiments (Sections 6.3 to 6.5), this method was used.

Table 4. Performance of the Proposed IE Method with Different Transfer Learning Strategy Combinations

Strategy combination	Feature-based transfer learning strategy	Model-based transfer learning strategy	Precision[1]	Recall[1]	F1 measure[1]
SC1	None	Two-stage training	79.7%	80.5%	80.1%
SC2	None	Alternating training	87.0%	87.5%	87.2%
SC3	Trainable pretrained word embeddings	Two-stage training	83.3%	84.0%	83.6%
SC4	**Trainable pretrained word embeddings**	**Alternating training**	**93.1%**	**92.9%**	**93.0%**
SC5	Fixed pretrained word embeddings	Two-stage training	83.4%	83.9%	83.7%
SC6	Fixed pretrained word embeddings	Alternating training	90.0%	90.5%	90.2%

[1]Bolded font indicates the highest performance.

6.3 Comparison of the performances of the proposed and baseline methods

To evaluate the effect of using deep neural networks and leveraging source-domain training data through transfer learning strategies on the extraction performance, the proposed IE method was compared to the linear CRF as a baseline. Linear CRF was selected because it has achieved the state-of-the-art performance for shallow IE in the AEC domain (e.g., [4]). Two linear CRF baseline models were constructed for performance comparison, one with word embeddings as features (Baseline 1) and another with both word embeddings and POS tags (Baseline 2). As shown in Table 5, compared to the baseline methods, the proposed IE method achieved higher performance, with an average increase of 9.6% in precision (14.2% for Baseline 1 and 4.9% for Baseline 2), 9.8% in recall (14.5% for Baseline 1 and 5.0% for Baseline 2), and 9.4% (14.4% for Baseline 1 and 4.4% for Baseline 2) in F1 measure.

Table 5. Performance of the Proposed IE Method Compared to the Baseline

Deep information extraction method	Precision[1]	Recall[1]	F1 measure[1]
Proposed IE method (using deep neural networks)	**93.1%**	**92.9%**	**93.0%**
Baseline 1 (using linear conditional random fields + word embeddings)	78.9%	78.4%	78.6%
Baseline 2 (using linear conditional random fields + word embeddings + part-of-speech tags)	87.9%	88.6%	88.2%

[1]Bolded font indicates the highest performance.

6.4 Performance of the proposed method on different types of regulatory documents

To evaluate the ability of the proposed IE method to extract syntactic and semantic information elements from regulatory documents that have different syntactic and semantic characteristics, the trained IE model was tested using building-code sentences from three different types of regulatory documents: the IBC, IECC, and ADA Standards, as shown in Table 6. The proposed IE method achieved consistent performance across the three types of documents, based on the three metrics, indicating that the method has high flexibility and scalability. As shown in Fig. 7, compared to the baseline methods, the proposed IE method achieved higher performance across the three types of documents. For IBC, the average increase is 11.5% in precision (17.3% for

Baseline 1 and 5.6% for Baseline 2), 11.3% in recall (17.1% for Baseline 1 and 5.5% for Baseline 2), and 11.1% (17.3% for Baseline 1 and 4.9% for Baseline 2) in F1 measure. For IECC, the average increase is 8.8% in precision (11.7% for Baseline 1 and 5.8% for Baseline 2), 8.2% in recall (12.6% for Baseline 1 and 3.7% for Baseline 2), and 8.5% (12.2% for Baseline 1 and 4.8% for Baseline 2) in F1 measure. For ADA, the average increase is 8.1% in precision (12.2% for Baseline 1 and 3.9% for Baseline 2), 8.3% in recall (12.6% for Baseline 1 and 3.9% for Baseline 2), and 8.2% (12.4% for Baseline 1 and 3.9% for Baseline 2) in F1 measure.

Table 6. Deep Information Extraction Performance Across Different Types of Regulatory Documents

Type of regulatory document	Deep information extraction method	Precision[1]	Recall[1]	F1 measure[1]
International Building Code	**Proposed method**	**94.9%**	**95.2%**	**95.1%**
	Baseline 1	77.6%	78.1%	77.8%
	Baseline 2	89.3%	89.7%	90.2%
International Energy Conservation Code	**Proposed method**	**87.3%**	**86.8%**	**87.1%**
	Baseline 1	75.6%	74.2%	74.9%
	Baseline 2	81.5%	83.1%	82.3%
Americans with Disabilities Act Standards	**Proposed method**	**95.1%**	**94.7%**	**94.9%**
	Baseline 1	82.9%	82.1%	82.5%
	Baseline 2	91.2%	90.8%	91.0%

[1]Bolded font indicates the highest performance.

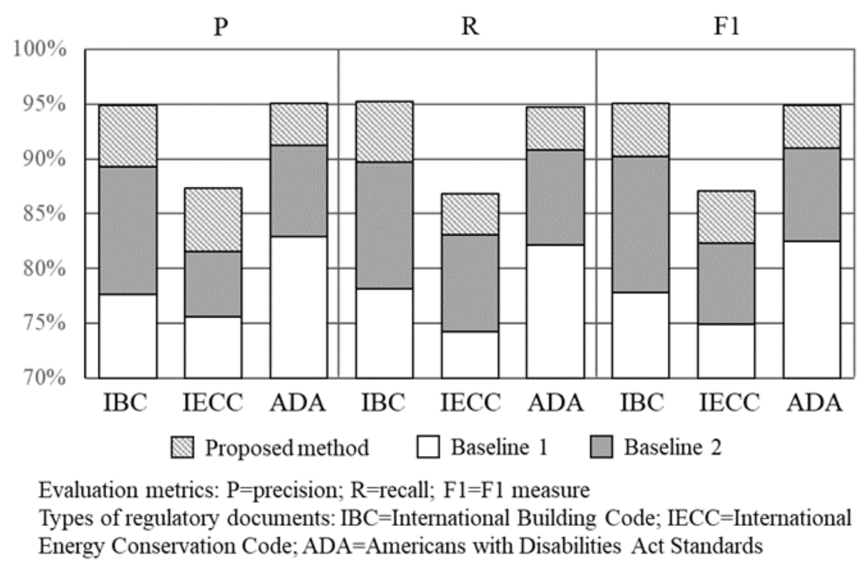

Fig. 7. Comparison of Deep Information Extraction Performance Across Different Types of Regulatory Documents

6.5 Performance of the proposed method on building-code sentences of different levels of computability

To evaluate the ability of the proposed IE method to extract syntactic and semantic information elements from different types of sentences, the trained IE model was tested using building-code sentences with different computability levels. Three different types of sentences were used for comparative evaluation, as shown in Table 7: moderately high, moderately low, and low computability, which are the top three types of sentences in terms of computability that appear most frequently in building codes (e.g., they account for 22%, 39%, and 23% of a corpus of sentences from IBC and its amendments, respectively) [49]. Sentences of moderately high computability have relatively simple syntactic and semantic structures (e.g., consisting of relatively short noun phrases, verb phrases, and preposition phrases at the sentence-level, or having simple or no restrictions). For example, "spacing of braced wall lines shall not exceed 35 feet on center in both the longitudinal and transverse directions in each story" has moderately high computability. Sentences of moderately low computability have relatively complex syntactic and semantic structures (e.g., consisting of relatively long noun phrases, verb phrases, and preposition phrases at the sentence-level, or having one recursive restriction). For example, "openings between the Group S-2 enclosed parking garage and Group S-2 open parking garage, except exit openings, shall not be required to be protected" has moderately high computability. Sentences of low computability have very complex syntactic and semantic structures (e.g., consisting of very long noun phrases, verb phrases, and preposition phrases at the sentence-level, or having multiple recursive restrictions). For example, "where exterior walls serve as a part of a required fire-resistance-rated shaft or exit enclosure, or separation, such walls shall comply with the requirements of Section 705 for exterior walls and the fire-resistance-rated enclosure or separation requirements shall not apply" has low computability.

The proposed method achieved consistent performance across the three types of building-code sentences, based on the three metrics, indicating that the method has high flexibility and scalability. Also, all three selected types of sentences have hierarchical complex structures [3,49], indicating that the method is able to deal with complex building-code syntactic and semantic structures. As shown in Fig. 8, compared to the baseline methods, the proposed IE method achieved higher performance across sentences with all three levels of computability. For moderately high computability, the average increase is 5.3% in precision (7.5% for Baseline 1 and 3.1% for Baseline 2), 5.2% in recall (8.1% for Baseline 1 and 2.2% for Baseline 2), and 5.2% (7.8% for Baseline 1 and 2.6% for Baseline 2) in F1 measure. For moderately low computability, the average increase is 8.4% in precision (13.3% for Baseline 1 and 3.4% for Baseline 2), 7.3% in recall (12.4% for Baseline 1 and 2.2% for Baseline 2), and 7.9% (12.9% for Baseline 1 and 2.8% for Baseline 2) in F1 measure. For low computability, the average increase is 12.6% in precision (18.0% for Baseline 1 and 7.2% for Baseline 2), 12.8% in recall (18.9% for Baseline 1 and 6.6% for Baseline 2), and 12.7% (18.5% for Baseline 1 and 6.9% for Baseline 2) in F1 measure. Both the proposed method and the baseline methods achieved high performance on sentences with moderately high computability, because they have relatively simple syntactic and semantic structures that are relatively easy to be captured by the models used in both methods. However, for sentences with low computability, the proposed method outperformed the baseline methods significantly, because they have relatively complex syntactic and semantic structures, especially long and recursive ones, which are better captured by the model used in the proposed method.

Table 7. Deep Information Extraction Performance for Building-Code Sentences with Different Computability Levels

Computability of building-code sentences	Deep information extraction method	Precision[1]	Recall[1]	F1 measure[1]
	Proposed method	**95.2%**	**93.8%**	**94.5%**
Moderately high	Baseline 1	87.7%	85.7%	86.7%
	Baseline 2	92.1%	91.6%	91.9%
	Proposed method	**93.8%**	**93.5%**	**93.7%**
Moderately low	Baseline 1	80.5%	81.1%	80.8%
	Baseline 2	90.4%	91.3%	90.9%
	Proposed method	**91.6%**	**92.1%**	**91.9%**
Low	Baseline 1	73.6%	73.2%	73.4%
	Baseline 2	84.4%	85.5%	85.0%

[1]Bolded font indicates the highest performance

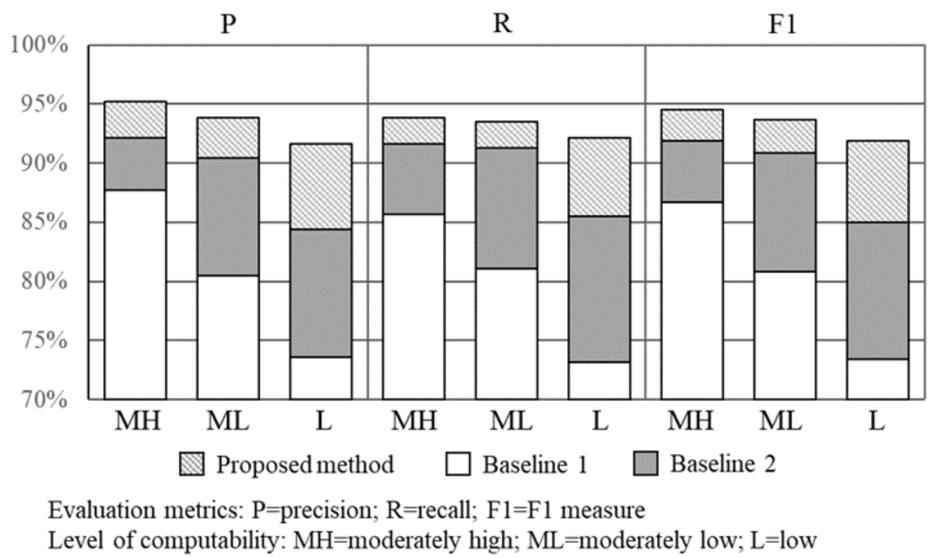

Fig. 8. Comparison of Deep Information Extraction Performance for Building-Code Sentences with Different Computability Levels

6.6 *Error analysis*

An error analysis was conducted to investigate the sources of errors and identify potential directions for performance enhancement in the future. To analyze the extraction errors, the confusion matrix (Fig. 9) was generated. Three main types of errors were identified based on the experimental results. First, the proposed approach had errors when dealing with multiword expressions, which consist of multiple words and function as individual syntactic and semantic units, especially those including prepositions. For example, the words in the multiword expression “means of egress” should have been annotated with a single semantic information element – a subject, but instead it was annotated with a subject, a syntactic unit, and another subject. In future work, a multiword expression list for the AEC domain could be integrated into the proposed method. Second, the proposed method performed relatively lower on extracting compliance checking attributes and references compared to other types of semantic and syntactic information elements, as shown in the confusion matrix. For example, the “required insulation” in “the requirement insulation for roof or ceiling assemblies” should have been extracted as a compliance checking attribute, but was misextracted as a subject. The “U-factor and SHGC requirements” should have been extracted together as a reference, but the “U-factor” was misextracted separately as a subject. Also, “Group R-1”, which means the first residential group in the IBC use and occupancy classification, was mistakenly extracted as part of a subject instead of a compliance checking attribute. In the future, additional input layers could be added to capture syntactic and semantic patterns that are discriminative in distinguishing subjects from compliance checking attributes and references. Third, the proposed method performed relatively lower on the IECC compared to other types of regulatory documents. The lower performance results from the relatively low amount of target-domain training data built using IECC sentences. In the future, more experiments are needed to evaluate the ability of the proposed method to scale to different types of regulatory documents when the amount of training data changes.

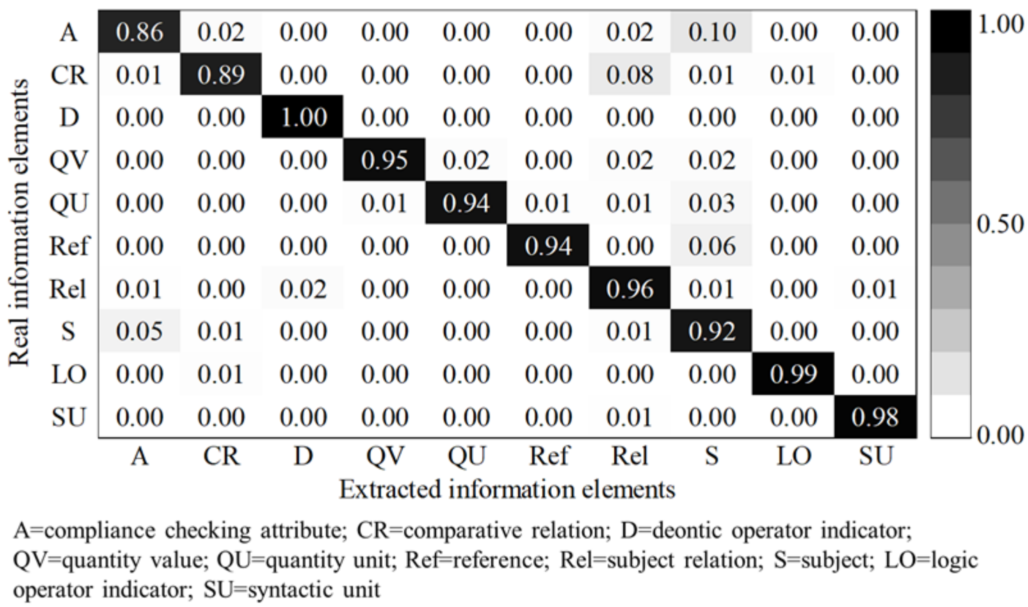

Fig. 9. Confusion matrix for semantic and syntactic information elements.

7 Contribution to the body of knowledge

This paper contributes to the body of knowledge on two levels. On a methodological level, the paper offers a new method that integrates deep learning, transfer learning strategies, and both target-domain and general-domain data to fully automatically extract semantic and syntactic information elements from regulatory documents for supporting ACC in the AEC domain. The proposed approach improves the methodology of information extraction in three primary ways. First, it is the first effort to use a deep learning-based method to fully automatically extract semantic and syntactic information elements from regulatory documents in the AEC domain for supporting fully automated compliance checking. Second, it leverages both general-domain and AEC-specific training data through transfer learning strategies to improve the performance, flexibility, and scalability of the proposed deep IE method. The experimental results indicate that the transfer learning strategies could greatly impact the IE performance. Third, the deep neural network architectures and transfer learning strategies used in the proposed deep IE method are adaptable to other types of text analytics tasks in the AEC domain such as requirement classification and semantic parsing.

On a practical level, the paper contributes to the body of knowledge in two ways. First, the paper proposes a set of semantic and syntactic information elements to facilitate the representation of building-code requirements and the extraction of regulatory information for supporting building-code analytics and compliance checking, which was effective for various types of regulatory documents such as IBC, IECC, and ADA Standards. Second, the paper offers a trained, ready-to-use deep IE model that offers high extraction performance, with consistency across different types of building codes and across sentences with different levels of computability. Third, both the information elements and the deep IE model would help achieve full automation in ACC systems, including full automation in extraction and formalization of requirements/rules. Fully automated ACC would reduce code compliance errors and the time and cost associated with compliance checking, thereby bringing broad benefits to the construction industry such as reduced violations, enhanced resource efficiency, and faster permitting.

8 Conclusions and future work

This paper proposed a deep learning-based method that uses transfer learning strategies for deep information extraction from regulatory documents for supporting automated compliance checking in the AEC domain. A set of semantic and syntactic information elements for representing building-code requirements was proposed and used for deep IE from regulatory documents in the AEC domain. Two types of training data, target-domain and general-domain data, were prepared using text from multiple AEC regulatory documents and from the Penn Treebank, respectively. The deep neural network model consists of bidirectional LSTM and CRF layers, which were adopted as the base IE model. Four different feature-based and model-based transfer learning strategies were used to adapt the base model and train the model on both domain-specific and general-domain training data.

The proposed deep IE method was tested and evaluated using building-code sentences collected from three types of regulatory documents (i.e., IBC, IECC, and ADA Standards). Different combinations of transfer learning strategies were tested and compared, and the optimal combination was to use pretrained word embeddings to initialize the transfer feature information and use alternating training to transfer the model information. Average precision of 93.1%, recall of 92.9%, and F1 measure of 93.0% were achieved under the optimal hyperparameters and transfer learning strategies, indicating good extraction performance and outperforming the baseline linear CRF-based method. Also, the trained deep IE model performed consistently across different types of regulatory documents including IBC, IECC, and ADA Standards, and different types of building-code sentences in terms of computability.

In their future work, the authors plan to improve the proposed method and leverage the deep IE model in five directions. First, the deep neural network model could be improved to enhance the extraction performance. For example, other model architectures, such as the Transformer-based architectures (e.g., finetuning BERT and its variants), could be explored. Second, more transfer learning and semi-supervised learning strategies could be explored for leveraging large-scale, pattern-rich general-domain annotated data for solving IE problems in the AEC domain. Third, the performance and flexibility of the IE model could be further improved by increasing the diversity of both the domain-specific and general-domain data. For example, annotated data from other sources could be used with data pruning techniques or instance-based transfer learning strategies. Fourth, additional evaluation efforts could be conducted to further test the proposed method on other types of regulatory documents and requirements. Reproducibility of the performance results are expected. However, the results may show performance variations due to possible differences in the syntactic and semantic characteristics of the documents or requirements. More comparative evaluation could also be undertaken in the future, as publicly available benchmark datasets become more available in the AEC domain. Fifth, and most importantly, the authors will further implement the trained IE model in an ACC system. Our ultimate goal is to leverage machine learning and other artificial intelligence approaches to reach a level where we can automatically process the entire building code and represent it in a computable manner for fully ACC with minimal manual effort.

9 Acknowledgements

The authors would like to thank the National Science Foundation (NSF). This material is based on work supported by the NSF under Grant No. 1827733. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

10 References

[1] Hjelseth, E. and Nisbet, N., 2010, November. Exploring semantic based model checking. In Proceedings of the 2010 27th CIB W78 international conference (Vol. 54). https://www.researchgate.net/profile/Eilif-Hjelseth/publication/265821429_EXPLORING_SEMANTIC_BASED_MODEL_CHECKING/links/550dbd2c0cf27526109c293c/EXPLORING-SEMANTIC-BASED-MODEL-CHECKING.pdf (Aug. 01, 2020).

[2] Zhang, J. and El-Gohary, N.M., 2013. Semantic NLP-based information extraction from construction regulatory documents for automated compliance checking. Journal of Computing in Civil Engineering, 30(2), p.04015014. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000346.

[3] Zhou, P. and El-Gohary, N., 2017. Ontology-based automated information extraction from building energy conservation codes. Automation in Construction, 74, pp. 103-117. https://doi.org/10.1016/j.autcon.2016.09.004.

[4] Liu, K. and El-Gohary, N., 2017. Ontology-based semi-supervised conditional random fields for automated information extraction from bridge inspection reports. Automation in construction, 81, pp. 313-327. https://doi.org/10.1016/j.autcon.2017.02.003.

[5] Liu, G. and Guo, J., 2019. Bidirectional LSTM with attention mechanism and convolutional layer for text classification. Neurocomputing, 337, pp. 325-338. https://doi.org/10.1016/j.neucom.2019.01.078.

[6] Etzioni, O., Fader, A., Christensen, J., Soderland, S. and Mausam, M., 2011, July. Open information extraction: The second generation. In IJCAI, 11, pp. 3-10. https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-012.

[7] Fader, A., Soderland, S. and Etzioni, O., 2011, July. Identifying relations for open information extraction. In Proceedings of the 2011 conference on empirical methods in natural language processing, pp. 1535-1545. https://www.aclweb.org/anthology/D11-1142 (Aug 01, 2020).

[8] Del Corro, L. and Gemulla, R., 2013, May. Clausie: clause-based open information extraction. In Proceedings of the 22nd international conference on World Wide Web, pp. 355-366. https://doi.org/10.1145/2488388.2488420.

[9] Gutierrez, F., Dou, D., Fickas, S., Wimalasuriya, D. and Zong, H., 2016. A hybrid ontology-based information extraction system. Journal of Information Science, 42(6), pp. 798-820. https://doi.org/10.1177/0165551515610989.

[10] Chambers, N. and Jurafsky, D., 2011, June. Template-based information extraction without the templates. In Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies, pp. 976-986. https://www.aclweb.org/anthology/P11-1098 (Aug. 01, 2020).

[11] Valenzuela-Escárcega, M.A., Hahn-Powell, G., Surdeanu, M. and Hicks, T., 2015, July. A domain-independent rule-based framework for event extraction. In Proceedings of ACL-IJCNLP 2015 System Demonstrations, pp. 127-132. https://doi.org/10.3115/v1/P15-4022.

[12] Kluegl, P., Toepfer, M., Beck, P.D., Fette, G. and Puppe, F., 2016. UIMA Ruta: Rapid development of rule-based information extraction applications. Natural Language Engineering, 22(1), pp. 1-40. https://doi.org/10.1017/S1351324914000114.

[13] Zhou, G. and Su, J., 2002. Named entity recognition using an HMM-based chunk tagger. In proceedings of the 40th Annual Meeting on Association for Computational Linguistics, pp. 473-480. https://doi.org/10.3115/1073083.1073163.

[14] Li, Y., Bontcheva, K. and Cunningham, H., 2004, September. SVM based learning system for information extraction. In International Workshop on Deterministic and Statistical Methods in Machine Learning, pp. 319-339. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11559887_19.

[15] Finkel, J.R., Grenager, T. and Manning, C., 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In Proceedings of the 43rd annual meeting on association for computational linguistics, pp. 363-370. https://doi.org/10.3115/1219840.1219885.

[16] LeCun, Y., Bengio, Y. and Hinton, G., 2015. Deep learning. Nature. 521(7553), pp. 436. https://doi.org/10.1038/nature14539.

[17] Chung, J., Gulcehre, C., Cho, K. and Bengio, Y., 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. https://arxiv.org/abs/1412.3555.

[18] Greff, K., Srivastava, R.K., Koutník, J., Steunebrink, B.R. and Schmidhuber, J., 2016. LSTM: A search space odyssey. IEEE transactions on neural networks and learning systems, 28(10), pp. 2222-2232. https://doi.org/10.1109/TNNLS.2016.2582924.

[19] Clark, K., Luong, M.T., Manning, C.D. and Le, Q.V., 2018. Semi-supervised sequence modeling with cross-view training. https://arxiv.org/abs/1809.08370.

[20] Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K. and Dyer, C., 2016. Neural architectures for named entity recognition. https://arxiv.org/abs/1603.01360.

[21] Nguyen, T.H. and Grishman, R., 2015. Event detection and domain adaptation with convolutional neural networks. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), pp. 365-371. https://doi.org/10.3115/v1/P15-2060

[22] Stanovsky, G., Michael, J., Zettlemoyer, L. and Dagan, I., 2018. Supervised open information extraction. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pp. 885-895. https://doi.org/10.18653/v1/N18-1081.

[23] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L. and Polosukhin, I., 2017. Attention is all you need. https://arxiv.org/abs/1706.03762.

[24] Radford, A., Narasimhan, K., Salimans, T. and Sutskever, I., 2018. Improving language understanding by generative pre-training. https://www.cs.ubc.ca/~amuham01/LING530/papers/radford2018improving.pdf (Feb. 10, 2021).

[25] Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. https://arxiv.org/abs/1810.04805.

[26] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. and Le, Q.V., 2019. Xlnet: Generalized autoregressive pretraining for language understanding. https://arxiv.org/abs/1906.08237.

[27] Sanh, V., Debut, L., Chaumond, J. and Wolf, T., 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. https://arxiv.org/abs/1910.01108.

[28] Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P. and Soricut, R., 2019. Albert: A lite bert for self-supervised learning of language representations. https://arxiv.org/abs/1909.11942.

[29] Zhu, J., Xia, Y., Wu, L., He, D., Qin, T., Zhou, W., Li, H. and Liu, T.Y., 2020. Incorporating bert into neural machine translation. https://arxiv.org/abs/2002.06823.

[30] Yang, W., Xie, Y., Lin, A., Li, X., Tan, L., Xiong, K., Li, M. and Lin, J., 2019. End-to-end open-domain question answering with bertserini. https://arxiv.org/abs/1902.01718.

[31] Nogueira, R. and Cho, K., 2019. Passage Re-ranking with BERT. https://arxiv.org/abs/1901.04085.

[32] Zhang, R. and El-Gohary, N., 2019. A machine learning-based approach for building code requirement hierarchy extraction. In 2019 CSCE Annual Conference. https://par.nsf.gov/servlets/purl/10110925 (Aug. 01, 2020).

[33] Pan, Y. and Zhang, L., 2020. BIM log mining: Learning and predicting design commands. Automation in Construction, 112, p.103107. https://doi.org/10.1016/j.autcon.2020.103107.

[34] Bang, S. and Kim, H., 2020. Context-based information generation for managing UAV-acquired data using image captioning. Automation in Construction, 112, p.103116. https://doi.org/10.1016/j.autcon.2020.103116.

[35] Marcus, M., Santorini, B. and Marcinkiewicz, M.A., 1993. Building a large annotated corpus of English: The Penn Treebank. https://repository.upenn.edu/cis_reports/237/ (Aug. 01, 2020).

[36] Sang, E.F. and De Meulder, F., 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. https://arxiv.org/abs/cs/0306050.

[37] Carreras, X. and Màrquez, L., 2005, June. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In Proceedings of the ninth conference on computational natural language learning (CoNLL-2005), pp. 152-164. https://www.aclweb.org/anthology/W05-0620 (Aug. 01, 2020).

[38] Tan C., Sun F., Kong T., Zhang W., Yang C., Liu C., 2018. A Survey on Deep Transfer Learning. In: Kůrková V., Manolopoulos Y., Hammer B., Iliadis L., Maglogiannis I. (eds) Artificial Neural Networks and Machine Learning – ICANN 2018. ICANN 2018. Lecture Notes in Computer Science, vol 11141. Springer, Cham. https://doi.org/10.1007/978-3-030-01424-7_27.

[39] Dai, W., Yang, Q., Xue, G.R. and Yu, Y., 2007. Boosting for transfer learning. In Proceedings of the 24th international conference on Machine learning, pp. 193-200. https://doi.org/10.1145/1273496.1273521.

[40] Pennington, J., Socher, R. and Manning, C.D., 2014. Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp. 1532-1543. https://doi.org/10.3115/v1/D14-1162.

[41] Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. and Zettlemoyer, L., 2018. Deep contextualized word representations. https://arxiv.org/abs/1802.05365.

[42] Kim, H., Kim, H., Hong, Y.W. and Byun, H., 2018. Detecting construction equipment using a region-based fully convolutional network and transfer learning. Journal of computing in Civil Engineering, 32(2), p.04017082. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000731.

[43] Zhang, K., Cheng, H.D. and Zhang, B., 2018. Unified approach to pavement crack and sealed crack detection using preclassification based on transfer learning. Journal of Computing in Civil Engineering, 32(2), p.04018001. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000736.

[44] Yang, Z., Salakhutdinov, R., and Cohen, W. W., 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. https://arxiv.org/abs/1703.06345.

[45] Al Qady, M. and Kandil, A., 2010. Concept relation extraction from construction documents using natural language processing. Journal of construction engineering and management, 136(3), pp. 294-302. https://doi.org/10.1061/(ASCE)CO.1943-7862.0000131.

[46] Lee, J., Yi, J.S. and Son, J., 2019. Development of automatic-extraction model of poisonous clauses in international construction contracts using rule-based NLP. Journal of Computing in Civil Engineering, 33(3), p.04019003. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000807.

[47] Zhang, R. and El-Gohary, N., 2019. A machine learning approach for compliance checking-specific semantic role labeling of building code sentences. In Advances in informatics and computing in civil and construction engineering, pp. 561-568. Springer, Cham. https://doi.org/10.1007/978-3-030-00220-6_67.

[48] Kim, T. and Chi, S., 2019. Accident case retrieval and analyses: Using natural language processing in the construction industry. Journal of Construction Engineering and Management, 145(3), p.04019004. https://doi.org/10.1061/(ASCE)CO.1943-7862.0001625.

[49] Zhang, R., and El-Gohary, N., 2020. Clustering-based Approach for Building Code Computability Analysis. Journal of Computing in Civil Engineering. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000967.

[50] J.P. Pestian, L. Deleger, G.K. Savova, J.W. Dexheimer, I. Solti, Natural language processing—the basics, Pediatric Biomedical Informatics: Computer Applications in Pediatric Research, Springer, Netherlands, Dordrecht, 2012, pp. 149–172, http://dx.doi.org/10.1007/978-94-007-5149-1_9. ISBN 978-94-007-5149-1.

[51] Huang, Z., Xu, W., and Yu, K., 2015. Bidirectional LSTM-CRF models for sequence tagging. https://arxiv.org/abs/1508.01991.

[52] Zhai, C., and Massung, S., 2016. Text data management and analysis: a practical introduction to information retrieval and text mining, ACM, New York, USA. https://doi.org/10.1145/2915031. ISBN: 978-1-970001-17-4.