Zhang, R., and El-Gohary, N. (2019). “A machine learning-based approach for building code requirement hierarchy extraction;" Proceedings of the 7th CSCE International Construction Specialty Conference (jointly with Construction Research Congress); Laval, Greater Montreal Area, June 12-15, 2019, Canada.

CSCE Annual Conference

Growing with youth – Croître avec les jeunes

Laval (Greater Montreal)

June 12 - 15, 2019

A MACHINE LEARNING-BASED METHOD FOR BUILDING CODE REQUIREMENT HIERARCHY EXTRACTION

Zhang, R.[1,2], and El-Gohary, N.[1,3]
[1] University of Illinois at Urbana-Champaign, US
[2] rzhang65@illinois.edu
[3] gohary@illinois.edu

Abstract: Most of the existing automated code compliance checking (ACC) methods are unable to fully automatically convert complex building-code requirements into computer-processable forms. Such complex requirements usually have hierarchically complex clause and sentence structures. There is, thus, a need to decompose such complex requirements into hierarchies of much smaller, manageable requirement units that would be processable using most of the existing ACC methods. Rule-based methods have been used to deal with such complex requirements and have achieved high performance. However, they lack scalability, because the rules are developed manually and need to be updated and/or adapted when applied to a different type of building code. More research is, thus, needed to develop a scalable method to automatically convert the complex requirements into hierarchies of requirement units to facilitate the succeeding steps of ACC such as information extraction and compliance reasoning. To address this need, this paper proposes a new, machine learning-based method to automatically extract requirement hierarchies from building codes. The proposed method consists of five main steps: (1) data preparation and preprocessing; (2) data adaptation; (3) deep neural network model training for dependency parsing; (4) automated requirement segmentation and restriction interpretation based on the extracted dependencies; and (5) evaluation. The proposed method was trained using the English Treebank data; and was tested on sentences from the 2009 International Building Code (IBC) and the Champaign 2015 IBC Amendments. The preliminary results show that the proposed method achieved an average normalized edit distance of 0.32, a precision of 89%, a recall of 76%, and an F1-measure of 82%, which indicates good requirement hierarchy extraction performance.

1 INTRODUCTION

To reduce the time, cost, and errors of compliance checking, various automated code compliance checking (ACC) methods have been developed and implemented. Although these methods have achieved different levels of automation, representativeness, and accuracy, most of them are unable to fully automatically convert complex building-code requirements into computer-processable forms. Such complex requirements usually have hierarchically complex clause and sentence structures (e.g., nested syntactic and semantic structures, conjunctive and alternative obligations, multiple exceptions, etc.). There is, thus, a need to decompose such complex requirements into hierarchies of much smaller, manageable requirement units that would be processable using most of the existing ACC methods.

Only a limited number of research efforts have focused on decomposing complex requirement sentences into semantically related units. Most of these research efforts, which have used annotation-based and/or rule-based methods, require intensive human effort. Annotation-based methods require experts to read and

annotate the building-code sentences with a set of semantic labels that indicate the role played by the sentence fragment in the context of compliance checking [e.g., requirement, application, selection, and exception (Hjelseth and Nisbet 2011)]. Recent rule-based methods (Zhang and El-Gohary 2013; Zhou and El-Gohary 2017) deal with complex requirements, segmenting a complex requirement into smaller semantic information elements including essential ones, such as subject and compliance checking attributes, and secondary ones, such as restrictions and exceptions. These rule-based methods have achieved the state-of-the-art performance, and only require limited human effort. However, they still lack scalability, because the rules are developed manually by experts and need to be updated and/or adapted when applied to a different type of building code and/or when the building code goes through major updates. More research is, thus, needed to develop a scalable method to automatically convert complex requirements into hierarchies of requirement units to facilitate the succeeding steps of ACC such as deep information extraction, information transformation, data matching, and compliance reasoning.

To address this need, this paper proposes a new, machine learning-based method to automatically extract the requirement hierarchies from building codes, which consist of requirement units and restriction relationships between the units, based on syntactic dependencies. To identify those dependencies, a deep neural network model was trained on out-of-domain, large-scale annotated data, which were first adapted to the architectural, engineering, and construction (AEC) domain based on data similarity. The proposed method is composed of five main steps: (1) prepare and preprocess training and testing data from both outside of and within the AEC domain; (2) adapt the out-of-domain training data to the domain-specific task; (3) train a machine-learning model to automatically perform dependency parsing; (4) extract the requirement hierarchies by segmenting each requirement sentence into requirement units and interpreting the restriction relationships between the requirement units based on the dependencies; and (5) evaluate the proposed method by evaluating the requirement segmentation and restriction interpretation separately, using average normalized edit distance, and precision, recall, and F1-measure, respectively. The proposed method was trained using the English Treebank data from the computational linguistic domain, and tested on sentences selected from the 2009 International Building Code (IBC) and the Champaign 2015 IBC Amendments, with each converted into a requirement hierarchy.

2 BACKGROUND

2.1 Requirement Semantic Representation

Semantic representations of requirements aim to represent the natural language requirements in computer-processable forms. Different semantic representations of requirements have been developed in the AEC domain for the purpose of regulatory document analysis in the context of compliance checking. Hjelseth and Nisbet (2011) proposed semantic markups including requirement, applicability, selection and exception (RASE) to model requirements with constraints. Zhang and El-Gohary (2013) and Zhou and El-Gohary (2017) proposed a set of semantic information elements to represent the information elements of requirements for facilitating the extraction of information from regulatory documents for supporting automated compliance checking. The semantic information elements include essential ones (e.g., subject, compliance checking attribute, quantity value) and secondary ones (e.g., subject restriction, quantity restriction). The building-code requirements are converted into the semantic representations either manually (Hjelseth and Nisbet 2011; Solihin and Eastman 2016) or automatically using hand-crafted rules (Zhang and El-Gohary 2013; Zhou and El-Gohary 2017).

2.2 Dependency Parsing

Dependency parsing aims to analyze the syntactic structure of a sentence by extracting the dependencies between a head word and words that directly relate to the head word (Nivre and Scholz 2004). The dependencies can be classified as three types: argument, modifier, and function-word dependencies. Argument dependencies are dependencies between arguments (i.e., a verb, the subject or object of a verb). Modifier dependencies are dependencies between an argument and the clauses, phrases, or words that modify this argument. Function-word dependencies are dependencies between words that indicate grammatical relationships and have little lexical meanings (e.g., prepositions, pronouns) and other words. Figure 1 shows the dependencies between the words of an example building-code sentence. The

dependencies can be used to further interpret the semantic relationships between the words, phrases, and/or fragments of the sentences.

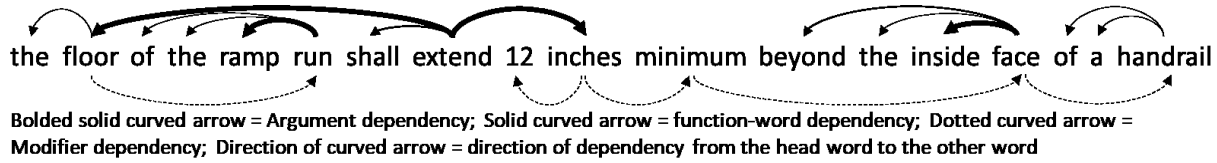

Figure 1: Dependencies between the words in an example sentence

The commonly used dependency parsing methods are transition-based. The transition-based methods update two sequences of words, the buffer (i.e., unprocessed words) and the stack (i.e., partially processed words), where initially the stack is empty and the buffer has all the words in the sentence (Nivre and Scholz 2004). One of the three transitions are performed – moving the word in the buffer to the stack, adding a dependency from the top word in the stack to the top word in the buffer and removing the top word from the stack, and adding a dependency from the top word in the buffer to the top word in the stack and moving the top word in the stack back to the buffer – until the buffer and the stack are empty. The transition and type of dependency are determined based on the features of the top words in the buffer and the stack.

2.3 Deep Learning

Deep learning methods use deep neural networks to automatically learn the patterns in large-scale training data (LeCun et al. 2015). Compared to traditional machine-learning methods, deep learning methods can achieve better performance, with least or no manual feature engineering. Deep neural networks have been used in the AEC domain recently, mainly for computer-vision tasks. For example, convolutional neural networks have been used for several tasks such as equipment detection (Kim et al. 2017), activity recognition (Luo et al. 2018), and crack/damage detection (Gulgec et al. 2019). However, no deep neural networks have been used in regulatory document analysis. To prevent overfitting problems, the training of deep neural networks requires a large, annotated dataset. However, the sizes of training data in the AEC domain are typically small (e.g., hundreds) due to the high cost of annotating the training data. One solution is to use a large-scale training dataset from another domain and adapt it to the target domain (e.g., AEC domain) by methods such as similarity-based data adaptation (Bhatt et al. 2015) and transfer learning (Yang et al. 2017). Such adaptation aims to adapt the syntactic and semantic features of the out-of-domain data so they become similar to those of the domain data, which could potentially increase the performance of the machine-learning model.

3 PROPOSED REQUIREMENT HIERARCHY MODELING

The paper proposes a new, hierarchical semantic representation to model complex requirements with restrictions and exceptions. A requirement hierarchy aims to represent a building-code requirement sentence in a hierarchical structure that consists of several requirement units and the relationships between the requirement units. Each requirement unit describes a requirement or condition on a subject and/or a compliance checking attribute. It may or may not include other essential semantic information elements (Zhang and El-Gohary 2013) such as quantitative relation, comparative relation, quantity value, quantity unit, and deontic operator indicator); and may not include any secondary semantic information elements such as restrictions and exceptions. Instead, a restriction or exception is represented as a separate, but related, requirement unit. A requirement unit is, thus, easily processable by most of the existing semi-automated or automated compliance checking methods and applications. There are three types of relationships between requirement units: conjunction (i.e., two units conjoined/disjoined by “and” and “or”), exception, and restriction. Restrictions can be further classified as subject restriction, compliance checking restriction, quantity restriction, and whole-unit restriction, depending on which requirement unit or semantic information element is being restricted/constrained. Conjunction and exception relationships can be directly

identified using function words, whereas restriction relationships cannot be easily identified. This paper, thus, focuses on restriction relationships.

The proposed method is designed to extract a requirement hierarchy given a building-code sentence, by first segmenting the sentence into units, and then interpreting the restriction relationships between the units. Figure 2 shows an example building-code sentence and the corresponding requirement hierarchy. In this example, the whole sentence is modeled as one hierarchy consisting of five requirement units. Each unit contains at least one subject and/or compliance checking attribute, as shown in Figure 2. Unit 4 (RU 4) is the restriction of Unit 3 (RU 3), and Units 2 and 3 (RU 2 and RU 3) are the restrictions of Unit 1 (RU 1), which is the main unit in the requirement hierarchy.

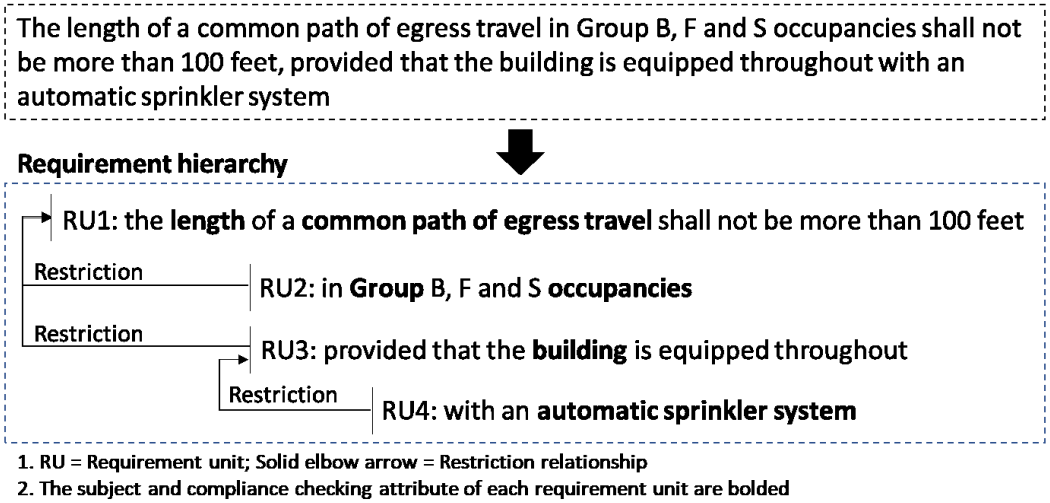

Figure 2: An example of requirement hierarchy modeling

4 PROPOSED MACHINE LEARNING-BASED REQUIREMENT HIERARCHY EXTRACTION METHOD

The proposed method consists of five main steps, as illustrated in Figure 3: data preparation and preprocessing, similarity-based training data pruning, dependency parsing model training, dependency-based requirement hierarchy extraction, and method evaluation.

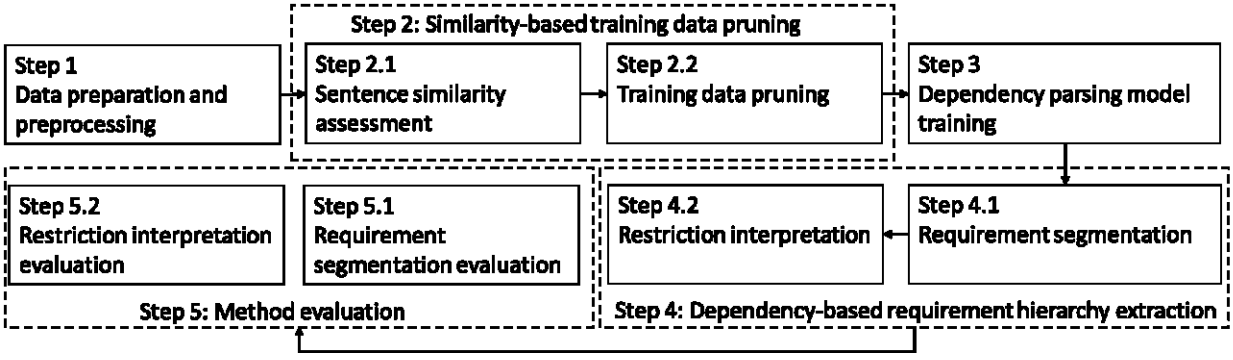

Figure 3: Proposed machine learning-based requirement hierarchy extraction method

4.1 Data Preparation and Preprocessing

Two types of data – out-of-domain and AEC-domain data – were prepared for training and testing. For out-of-domain data, the English Treebanks from the Universal Dependencies (Nivre 2018) were used for

training. The English Treebanks consist of 22,000 sentences collected from multiple domains (e.g., news, Wikipedia, social media) that are annotated with dependencies between words. The English Treebanks are large in scale, rich in syntactic and semantic patterns, and already annotated, which makes them suitable for training the deep neural network model for dependency parsing.

For AEC-domain data, a dataset of 6,000 sentences were collected from the 2009 International Building Code (IBC) and the Champaign 2015 IBC Amendments. The dataset was randomly split into two datasets: 5,850 sentences for training data pruning (Step 2) and 150 sentences for testing (Step 5). For annotating the testing dataset, the sentences were manually converted into requirement hierarchies, by first segmenting each sentence into requirement units and then annotating pairs of requirement units with restriction relationships. The AEC-domain data were then preprocessed using four steps: tokenization, lowercasing, stemming, and part-of-speech (POS) tagging.

4.2 Similarity-Based Training Data Pruning

4.2.1 Sentence Similarity Assessment

Data similarity assessment aims to compute the similarity between the out-of-domain data and the AEC-domain data. The proposed similarity between two sentences is defined as the cosine similarity between the embeddings of the two sentences. The sentence embedding is a vector representation of the semantics and syntactics of the sentence, and is formed by concatenating the average word embedding and the average POS-tag embedding of the sentence. The average word embedding is the average of the word embeddings of all the words in the sentence; and the average POS-tag embedding is the average of the POS-tag embeddings of all the POS tags corresponding to the words in the sentence. A word or POS-tag embedding is a vector representation of the word or POS tag in a specific context (e.g., building code) (Mikolov et al. 2013). Both word and POS-tag embeddings were trained using the corpus of 5,850 building-code sentences, using the word2vec algorithm by Gensim (Rehurek and Sojka 2010) built in Python.

4.2.2 Training Data Pruning

The out-of-domain training data were pruned based on data similarity. For each training sentence, the average of the similarities to all of the testing sentences were calculated. All the training sentences were then ranked based on the average similarities. If a training sentence ranks below a percentage threshold, the training sentence is pruned. Different pruning thresholds were tested and compared in terms of final requirement hierarchy extraction performance.

4.3 Dependency Parsing Model Training

The deep neural network architecture proposed by Kiperwasser and Goldberg (2016) was adopted for dependency parsing, which consists of three main layers: the embedding layer, the long short term memory (LSTM) layer, and the multi-layer perceptron (MLP) layer. The embedding layer aims to create the word and POS-tag embeddings for each word in the training sentences. The LSTM layer aims to compute the feature values using the word and POS-tag embeddings of the current word and context words. Given a transition-based dependency parsing status (i.e., the buffer and the stack), the MLP layer aims to compute the scores for all the three possible transitions and all the possible types of dependencies, using the output feature values of the LSTM layer. The transition and/or dependency with the highest score is selected. The basic model architecture is shown in Figure 4. The hyperparameters of the model were tuned (e.g., the number of LSTM layers) for improving the performance of dependency parsing.

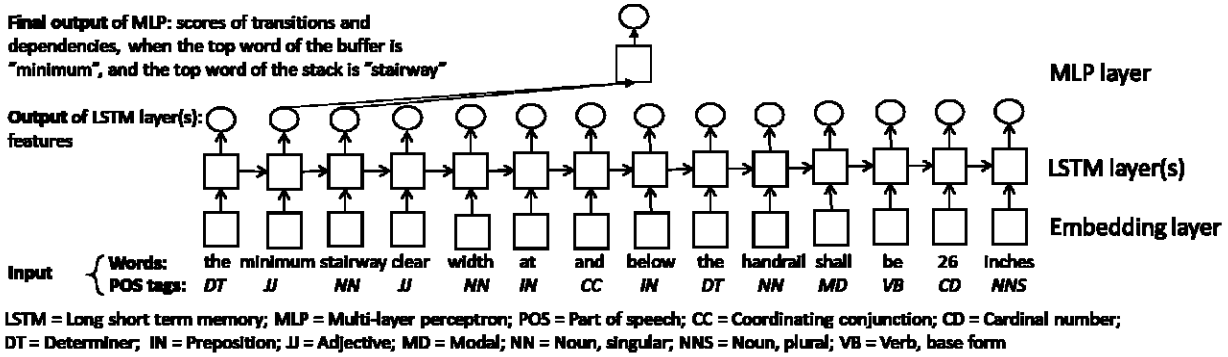

Figure 4: The basic architecture of the deep neural network model for dependency parsing

4.4 Dependency-based Requirement Hierarchy Extraction

The requirement hierarchy was then extracted from each sentence based on the dependencies, using two steps: requirement segmentation and restriction interpretation.

4.4.1 Requirement Segmentation

A requirement sentence is first segmented into several requirement units based on the word dependencies resulting from Step 3. Two simple rules were developed for requirement segmentation. First, any two words that have argument or function-word dependencies should belong to the same requirement unit. Second, any two words that have modifier dependencies are the segmentation boundary for two requirement units. The second rule has two exceptions: (1) one of the words is a cardinal number, because it acts as the quantity value in the requirement unit; and (2) one of the words is an adjective because it could be part of a subject or compliance checking attribute (e.g., “nonload-bearing” as an adjective modifying “wall” in the subject “nonload-bearing wall”), or could indicate a comparative relation (e.g., “minimum” as an adjective modifying the quantity value in the sentence “the floor shall extend 12 inches minimum”) in a requirement unit.

Figure 5 shows an example of requirement segmentation. In the example, the dependencies between the word pairs “room” and “openings”, “protectives” and “protected”, “having” and “protectives” are modifier dependencies, and thus those pairs are the segmentation boundaries. The dependencies between word pairs “3/4-hour” and “rating” and “less” and “3/4-hour” are modifier dependencies but those words are either cardinal numbers (i.e. “3/4-hour”) or adjectives (i.e., “less”), and thus should not belong to different requirement units. The rest of the dependencies are either argument (e.g., the dependency between “openings” and “protected”) or function-word dependencies (e.g., the dependency between “the” and “room”), and thus the words linked by those dependencies shall not be the segmentation boundaries.

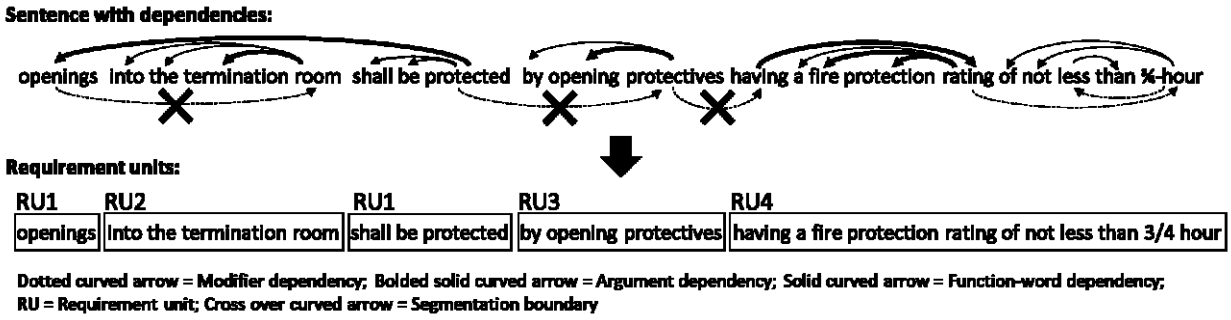

Figure 5: An example of dependency-based requirement segmentation

Zhang, R., and El-Gohary, N. (2019). “A machine learning-based approach for building code requirement hierarchy extraction;" Proceedings of the 7th CSCE International Construction Specialty Conference (jointly with Construction Research Congress); Laval, Greater Montreal Area, June 12-15, 2019, Canada.

4.4.2 Restriction Interpretation

The restriction relationships between the requirement units were then interpreted based on their modifier dependencies. A modifier dependency between a pair of words belonging to two requirement units indicates a restriction relationship between the two units. The direction of the modifier dependency (i.e., from a word to its head word) indicates the direction of the restriction relationship (i.e., a unit is the restriction of the other unit). Figure 6 shows an example of restriction interpretation. In the example, there is a modifier dependency between the word “room” and its head word “openings”. Thus, Unit 2 (RU2), which includes the word “room”, is the restriction of Unit 1 (RU1), which includes the head word “openings”. Similarly, the modifier dependencies between the word “protectives” and its head word “protected”, and between the word “having” and its head word “protectives”, indicate that RU3 is the restriction of RU1, and that RU4 is the restriction of RU3, respectively.

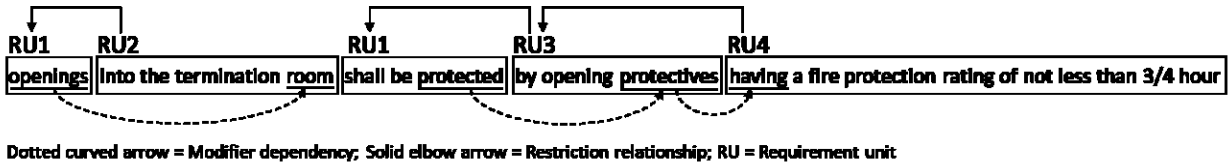

Figure 6: An example of dependency-based restriction interpretation

4.5 Method Evaluation

The performances of requirement segmentation and restriction interpretation were evaluated separately.

4.5.1 Requirement Segmentation Evaluation

The average normalized edit distance was used to evaluate the requirement segmentation performance. The normalized edit distance between two requirement units is defined as the minimum number of operations (i.e., insertion, deletion, and substitution) needed to transform one unit to the other, normalized by the lengths of the two units (Marzal and Vidal 1993). The average normalized edit distance is computed as the average of the minimum normalized edit distances from the requirement unit in the testing data to the requirement units generated by the requirement segmentation step. This metric ranges from 0 to 1. The lower the distance, the more similar the requirement units resulting from the proposed method to the requirement units in the gold standard, and the better the requirement segmentation performance; and vice versa.

4.5.2 Restriction Interpretation Evaluation

Three metrics were used to evaluate the restriction interpretation performance: precision (Eq. 1), recall (Eq. 2), and F1-measure (Eq. 3), where TP is the number of true positives (i.e., number of pairs of requirement units correctly labeled as having restriction relationship), FP is the number of false positives (i.e., number of pairs of requirement units incorrectly labeled as having restriction relationship), and FN is the number of false negatives (i.e., number of pairs of requirement units not labeled as having restriction relationship but should have been) (Zhai and Massung 2016). All the three metrics range from 0 to 1. The higher the metrics, the better the restriction interpretation performance. The testing of the restriction interpretation step was conducted using the gold standard units to evaluate each step (requirement segmentation and restriction interpretation) separately and independently.

[1] $\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}}$

[2] $\text{Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}}$

[3] $\text{F1} = \frac{2\times\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}$

Zhang, R., and El-Gohary, N. (2019). “A machine learning-based approach for building code requirement hierarchy extraction;" Proceedings of the 7th CSCE International Construction Specialty Conference (jointly with Construction Research Congress); Laval, Greater Montreal Area, June 12-15, 2019, Canada.

5 PRELIMINARY EXPERIMENTAL RESULTS

5.1 Performance Results

The performance results of requirement segmentation and restriction interpretation are summarized in Tables 1 and 2. The average normalized distance is 0.32, indicating good requirement segmentation. The precision, recall, and F1-measure for restriction interpretation is 89%, 76%, and 82%, respectively, indicating good requirement hierarchy extraction.

Different pruning thresholds were tested, including 10% (slight pruning), 50% (halving), and 75% (significant pruning), as shown in Table 1. The optimal performance for requirement segmentation and restriction interpretation was achieved at 10% (slight pruning) and 50% (halving) pruning thresholds, respectively. But, the differences were small. Compared to not pruning the training data, the slight pruning decreased the average normalized distance by 0.01, and halving increased the precision, recall, and F1-measure by 2%, which indicates the potential of pruning in improving the performance of the proposed method. However, when 75% of the data were pruned, the performances of the proposed method started to decrease, possibly because the size of the training data shrank to a point where the dependency parsing model suffered from overfitting.

Two dependency parsing algorithms were tested for comparative evaluation: the deep neural network model (see Section 4.3) and the neural network dependency parsing model by the Stanford CoreNLP (Chen and Manning 2014), as shown in Table 2. The deep learning model for dependency parsing achieved better performance for both requirement segmentation (lower average normalized edit distance) and restriction interpretation (higher precision, recall, and F1-measure). But, the differences were small. Compared to using the traditional neural network algorithm for the dependency parsing step, using the deep neural network algorithm decreased the average normalized edit distance by 0.13, and increased the precision, recall, and F1-measure by 2%. More testing is needed in future work to further verify these results and study the statistical and practical significances of these performance differences.

Table 1: Performance of the proposed method with different pruning thresholds (and using deep neural network model for dependency parsing)

Pruning method	Pruning threshold	Requirement segmentation[1]	Restriction interpretation[1]		
		Average normalized edit distance	Precision	Recall	F1-measure
No pruning	0%	0.33	87%	74%	80%
Slight pruning	10%	**0.32**	88%	74%	80%
Halving	50%	0.34	**89%**	**76%**	**82%**
Significant pruning	75%	0.34	86%	74%	80%

[1]Bolded font indicates highest performance.

Table 2: Performance of the proposed method with different dependency parsing algorithms (and using 10% pruning for requirement segmentation and 50% pruning for restriction interpretation)

Machine learning-based dependency parsing algorithm	Requirement segmentation[1]	Restriction interpretation[1]		
	Average normalized edit distance	Precision	Recall	F1-measure
Deep neural network algorithm	**0.32**	**89%**	**76%**	**82%**
Traditional neural network algorithm	0.45	87%	74%	80%

[1]Bolded font indicates highest performance.

5.2 Error Analysis

Two main types of errors were identified based on the experimental results. For requirement segmentation, the proposed method had errors when dealing with sentences that contain multiword expressions that are

made up of multiple words and act as a single syntactic and/or semantic unit (Sag et al. 2002), such as “in accordance with” and “means of egress”. The words in a fixed expression shall belong to a single requirement unit, while the proposed method segmented them into multiple units. In future work, a gazetteer list of fixed expressions used in the AEC domain could be added to the requirement segmentation step. For restriction interpretation, the proposed method had errors when dealing with sentences that have a mixed structure of preposition phrases and clauses (e.g., “the compartment shall extend through the highest level of exit discharge serving the underground portions”). The proposed method linked the requirement unit corresponding to the modifier clause (e.g., “serving the underground portions”) to the unit corresponding to the main clause (e.g., “the compartment shall extend through the highest level”) mistakenly, rather than to the unit corresponding to the preposition phrase (e.g., “of exit discharge”). In future research, more sentences that have this type of structure could be used for training the dependency parsing model.

6 CONCLUSIONS AND FUTURE WORK

This paper proposed a new semantic representation for modeling requirement sentences as hierarchies of smaller units and a machine learning-based method for automatically extracting such requirement hierarchies from building-code sentences for supporting automated compliance checking. First, a similarity-based data pruning method was proposed to adapt the out-of-domain English Treebank data, which are large in scale and rich in syntactic and semantic patterns, to the AEC-domain task at hand. Second, a deep neural network-based dependency parsing model was trained using the adapted data. Third, the requirement hierarchies were extracted by segmenting the requirement sentences into units, and interpreting the restriction relationships between the units, based on the dependencies. The proposed method achieved an average normalized distance of 0.32, indicating good requirement segmentation performance; and achieved a precision of 89%, a recall of 76%, and an F1-measure of 82%, indicating good restriction interpretation performance.

This paper contributes to the body of knowledge in two primary ways. First, the paper proposed a new way to model the complex requirement sentences, which can be used to convert complex requirements into hierarchies of much simpler requirement units to facilitate automated machine learning-based processing of those complex requirements. Second, the results show that the dependencies can be used in analyzing the semantic relationships between different parts of the requirement sentences and can be used to automatically segment the requirements and extract the restriction relationships.

In their future work, the authors first plan to improve the requirement modeling by identifying different types of restrictions based on examining building-code sentences, in terms of the semantic information elements that are restricted (e.g., subject restriction, compliance checking attribute restriction, quantity restriction) and the content of the restriction (e.g., location restriction, time restriction, method restriction). Second, the authors will explore further ways to improve the performance of the proposed requirement hierarchy extraction method, including using more training data, exploring different data adaptation methods (e.g., different similarity measures), and integrating the use of domain ontology for restriction interpretation. Third, and most importantly, the authors plan to integrate the proposed requirement hierarchy extraction method with machine learning-based information extraction, with an aim to find a scalable method for automated compliance checking.

ACKNOWLEDGEMENTS

This publication is based upon work supported by the National Science Foundation under Grant No. 1827733.

REFERENCES

Bhatt, H.S., Semwal, D. and Roy, S. 2015. An iterative similarity based adaptation technique for cross-domain text classification. *19th Conference on Computational Natural Language Learning*, ACL, Beijing, China: 52-61.

Zhang, R., and El-Gohary, N. (2019). “A machine learning-based approach for building code requirement hierarchy extraction;" Proceedings of the 7th CSCE International Construction Specialty Conference (jointly with Construction Research Congress); Laval, Greater Montreal Area, June 12-15, 2019, Canada.

Chen D. and Manning C.D. 2014. A Fast and Accurate Dependency Parser Using Neural Networks. *2014 conference on empirical methods in natural language processing*, ACL, Doha, Qatar: 740-750.

Gulgec, N.S., Takáč, M. and Pakzad, S.N. 2019. Convolutional Neural Network Approach for Robust Structural Damage Detection and Localization. *Journal of Computing in Civil Engineering*, ASCE, **33**(3): p.04019005.

Hjelseth, E. and Nisbet, N. 2010. Exploring semantic based model checking. *2010 27th CIB W78 international conference*, CIB, Cairo, Egypt, **54**.

Kim, H., Kim, H., Hong, Y.W. and Byun, H. 2017. Detecting construction equipment using a region-based fully convolutional network and transfer learning. *Journal of Computing in Civil Engineering*, ASCE, **32**(2): p.04017082.

Kiperwasser, E. and Goldberg, Y. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, ACL, 4: 313-327.

LeCun, Y, Bengio, Y. and Hinton, G. 2015. Deep learning. *Nature,* Nature Research, **521**(7553): 436.

Luo, X., Li, H., Cao, D., Dai, F., Seo, J. and Lee, S. 2018. Recognizing Diverse Construction Activities in Site Images via Relevance Networks of Construction-Related Objects Detected by Convolutional Neural Networks. *Journal of Computing in Civil Engineering*, ASCE, **32**(3): p.04018012.

Marzal, A. and Vidal, E. 1993. Computation of Normalized Edit Distance and Applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE Computer Society, **15**(9): 926-932.

Mikolov, T., Chen, K., Corrado, G. and Dean, J. 2013. Efficient Estimation of Word Representations in Vector Space. arXiv preprint arXiv:1301.3781.

Nivre, J. “Current UD Languages – English Treebanks.” Universal Dependencies. November 15, 2018. https://universaldependencies.org/#download.

Nivre, J. and Scholz, M. 2004. Deterministic Dependency Parsing of English Text. *20th International Conference on Computational Linguistics*, ACL, Geneva, Switzerland: 64.

Rehurek, R. and Sojka, P. 2010. Software Framework for Topic Modelling with Large Corpora. LREC *2010 Workshop on New Challenges for NLP Frameworks*, ELRA, Valletta, Malta: 46-50.

Sag, I.A., Baldwin, T., Bond, F., Copestake, A. and Flickinger, D. 2002. Multiword expressions: A pain in the neck for NLP. *International Conference on Intelligent Text Processing and Computational Linguistics*, Springer, Berlin, German: 1-15.

Z. Yang, R. Salakhutdinov and Cohen, W.W. 2017. Transfer Learning for Sequence Tagging with Hierarchical Recurrent Networks. arXiv preprint arXiv:1703.06345.

Zhai, C. and Massung, S. 2016. *Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining*. ACM, New York, NY, USA.

Zhang, J. and El-Gohary, N. 2013. Semantic NLP-based Information Extraction from Construction Regulatory Documents for Automated Compliance Checking. *Journal of Computing in Civil Engineering*, ASCE, **30**(2): p.04015014.

Zhou, P. and El-Gohary, N. 2017. Ontology-Based Automated Information Extraction from Building Energy Conservation Codes. *Automation in Construction*, Elsevier BV, **74**: 103-117.