A Deep-Learning Method for Evaluating Semantically-Rich Building Code Annotations

Ruichuan Zhang, Nora El-Gohary
University of Illinois at Urbana-Champaign, USA
rzhang65@illinois.edu

Abstract. Existing automated code checking (ACC) methods require the extraction of requirements from building codes and the representation of these requirements in a computer-processable form. Although these methods have achieved different levels of performance, all of them are still unable to represent all types of building-code requirements. There is, thus, a need to enhance the semantic representations of building codes towards facilitating the representation of all requirements. To address this need, this paper first proposes a new approach to annotate and represent building-code sentences using requirement units that consist of semantic information elements and simple logic operators. To evaluate the proposed building-code annotation approach, this paper also proposes a new natural language generation (NLG)-based method for evaluating annotation quality. The proposed method consists of four steps: data preparation, data preprocessing, NLG model development and training, and sentence evaluation. Sentences from the International Code Council (ICC) building codes were used in the evaluation.

1. Introduction

To reduce the time, cost, and errors of compliance checking, various automated code compliance checking (ACC) systems have been developed and implemented. Most of the existing ACC systems require the extraction of requirements from building-code sentences and the conversion of the extracted requirements into a semantic representation. Semantic representations of building codes aim to represent the natural language requirements in computer processable forms. Different semantic representations of requirements have been developed in the construction domain for supporting compliance checking. Examples of semantic representations proposed in research efforts include the requirement, applicability, selection and exception (RASE) semantic markups proposed by Hjelseth and Nisbet (2011), the semantic information elements proposed by Zhang and El-Gohary (2013), and the visual language for compliance checking proposed by Preidel and Borrmann (2016). Examples of semantic representations used in commercial ACC applications include the predefined templates used in the Solibri Model Checker and the hardcoded building-requirements in the SMARTreview.

However, two knowledge gaps still exist. First, although existing ACC methods and applications have achieved different levels of automation, representativeness, and accuracy, they are still unable to represent all building-code requirements. For example, the RASE markups can be used to annotate nested clauses and exceptions in building-code sentences, but cannot represent "requirements that include alternatives and preferences" (Hjelseth and Nisbet 2011). Similarly, the predefined templates in Solibri Model Checker can only represent a limited number of requirements in a few building codes, such as the clearance requirements in the Standards for Accessible Design by Americans with Disabilities Act, and the dimension requirements in the International Building Code (IBC). There is, thus, a need to enhance the semantic representations or annotations of building codes towards facilitating the representation of all requirements. Second, most of the existing semantic representations were developed without sufficient evaluation in terms of their representativeness of building-code regulatory

information. There is a lack of well-defined quantitative metrics and methods for evaluating the semantic representations and/or annotations of building codes.

To address the first knowledge gap, this paper proposes a new approach to annotate and represent building-code sentences using requirement units that consist of semantic information elements and simple logic operators. To address the second gap and evaluate the proposed annotation approach, this paper then proposes a natural language generation (NLG)-based method for evaluating annotation quality, where the building-code annotations are first used to generate new building-code sentences, and then the comprehensibility of the generated sentences are evaluated. Two metrics were used to evaluate the comprehensibility of the generated building-code sentences: bilingual evaluation understudy (BLEU) and recall-oriented understudy for gisting evaluation (ROUGE).

2. Background

2.1 Natural Language Generation

Natural language generation (NLG) is the process of representing the semantic information contained in the input data – which could be in various forms such as tables, images, or formal languages – in the form of natural language for the purpose of information digestion and communication (Arria NLG 2020). NLG plays a core role in many intelligent systems and applications such as spoken dialogue systems (Wen et al. 2015), image captioning (You et al. 2016), and business intelligence dashboards (Arria NLG 2020).

NLG methods range from template, rule, and heuristic-based methods to recent deep learning-based methods. Template, rule, and heuristics-based methods rely on manually-developed templates and rules to fill in the templates. Examples of template-based systems in the construction domain include the web-based document management system proposed by Ryoo et al. (2010); and commercial software such as e-Specs (Avitru 2018) and BIMdrive (Digicon 2018), which use premade templates (e.g., templates based on the National Master Specifications) to facilitate the development and maintenance of construction specifications. Deep learning-based methods use deep neural networks to automatically learn the syntactic and semantic patterns in the input data, which are later used to generate new text (e.g., semantically conditioned deep neural network-based text generation approach by Wen et al. 2015).

2.2 Deep Learning

Deep learning methods use computational models such as deep neural networks to learn multiple levels of information representations from large-scale data (LeCun et al. 2015). Deep learning methods have drastically improved the state-of-the-art performance in the natural language processing domain, and have eliminated a lot of the manual effort in feature engineering compared to traditional machine learning methods. In the construction domain, deep learning methods have been used to solve text analysis problems such as building-code requirement extraction (Zhang and El-Gohary 2019), building information modeling log data mining (Pan and Zhang 2020), and regulatory document semantic analysis (Zhang and El-Gohary 2020).

The most commonly used deep neural networks for dealing with sequential data such as text data are recurrent neural networks (RNNs). RNNs have recurrent connections between neurons in the neural networks to cycle the input information in order to learn the patterns of sequential data (Mikolov et al. 2010). To tackle the shortcoming of original RNNs, which is being less

capable and less computationally efficient to learn from long-term dependencies in the sequential data, variants of RNNs have been designed and adopted such as long short term memories (LSTMs) and gated recurrent units (GRUs).

3. Proposed Approach to Annotate Building-Code Sentences for Supporting Automated Compliance Checking

This paper proposes a new approach to annotate building-code sentences using requirement units. Each requirement unit consists of semantic information elements and simple logic operators. The semantic information elements include the essential semantic information elements proposed by Zhang and El-Gohary (2013), as shown in Table 1, in addition to a new element, "subject relation", which describes a relation between two subjects in one requirement unit. The simple logic operators include conjunction (e.g., "and"), disjunction (e.g., "or"), and negation (e.g., "not"). Each requirement unit describes a requirement or condition on a subject and/or a compliance checking attribute, or a pair of subjects, and thus is easily processable by most of the existing semi-automated or automated compliance checking applications and systems. Figure 1 shows an example building-code sentence that is annotated with requirement units.

Table 1: Essential semantic information elements for representing requirements for compliance checking purposes (Zhang and El-Gohary 2013).

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 (e.g., width)
Quantitative relation	A term/phrase that defines the type of relation for the quantity (e.g., extend)
Deontic operator indicator	A term/phrase that indicates the deontic type of the requirement (i.e., obligation, permission, or prohibition)
Comparative relation	A term/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"

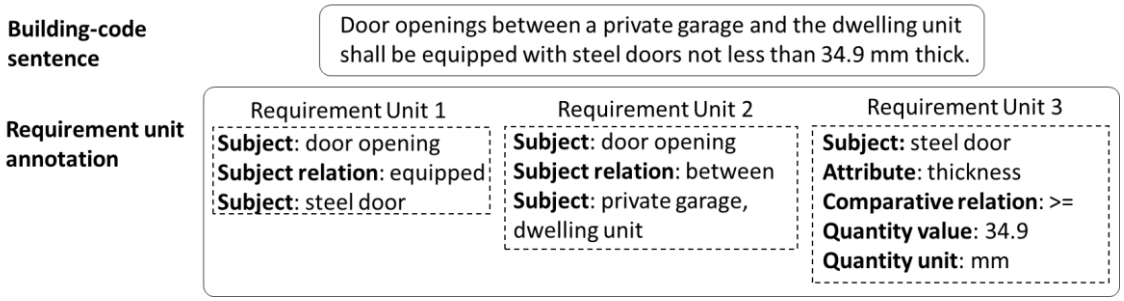

Figure 1: Example of Proposed Building-Code Sentence Annotations with Requirement Units

4. Proposed Deep Learning Natural Language Generation Method for Evaluating the Proposed Building-Code Annotation Approach

To evaluate the proposed building-code annotation approach, this paper proposes a deep learning natural language generation (NLG) method for evaluating annotation quality, where the annotations are first used to generate new sentences, and then the comprehensibility of the generated sentences are evaluated. The proposed method consists of four main steps (as per Figure 2): (1) preparing the training and testing data (i.e., pairs of natural language building-code sentence and the corresponding annotations); (2) preprocessing the data; (3) developing and training the deep learning model for generating building-code sentences; and (4) evaluating the comprehensibility of the generated building-code sentences using two sets of metrics: bilingual evaluation understudies (BLEU) and recall-oriented understudies for gisting evaluation (ROUGE).

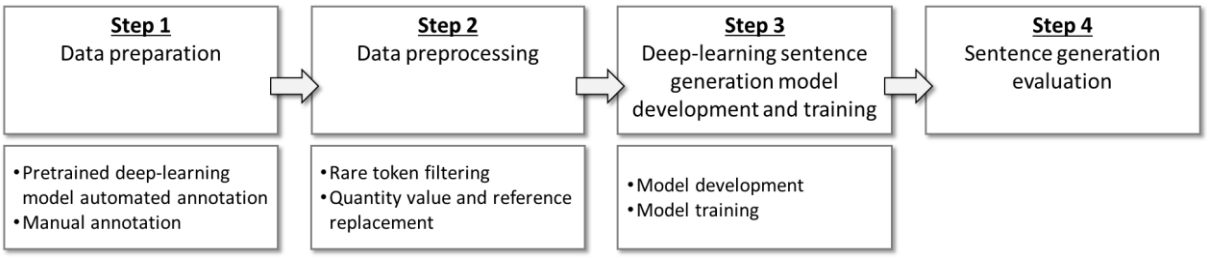

Figure 2: Proposed Deep Learning Natural Language Generation (NLG) Method for Evaluating the Proposed Building-Code Annotations

4.1 Data Preparation

Two datasets were used for training and testing the deep-learning building-code sentence generation model, which were annotated using two different methods. For training, to reduce the human effort, a relatively large dataset of sentences was collected and automatically annotated using a pretrained deep-learning model. For testing, a smaller dataset was collected and manually annotated by human annotators.

Automated Annotation of Training Data Using Pretrained Deep-Learning Model. A total of 80,000 sentences were collected from multiple regulatory documents including the International Energy Conservation Code, the International Residential Code, and the American National Standards for Accessible and Usable Buildings and Facilities. The pretrained deep information extraction model by Zhang and El-Gohary (2020) was used to automatically annotate the dataset with the proposed annotations.

Manual Annotation of Testing Data. A total of 100 sentences were collected from multiple chapters of the IBC 2009 and the Champaign 2015 IBC Amendments. The sentences were manually annotated by three different annotators.

4.2 Data Preprocessing

To further improve the quality of the training data and the robustness of the deep-learning sentence generation model, two data preprocessing steps were conducted: rare-token filtering, and quantity value and reference replacement.

Rare-Token Filtering. To reduce the effect of the noise in the training data, and to improve the model's ability to deal with words that are missing from the training data, rare tokens were

filtered. First, the token frequencies in the entire corpus of building-code sentences were counted. Then, the tokens with frequencies less than the threshold were replaced by the single missing-word token.

Quantity-Value and Reference Replacement. Quantity values share similar semantic and syntactic patterns, and so do references (e.g., the index of a chapter, table, or building code). Thus, two single tokens, "quantity value" and "reference", were used to replace tokens annotated as quantity values and tokens annotated as references, respectively, aiming to reduce the number of parameters to be learned in the sentence generation model and thus the possibility of overfitting.

4.3 Deep-Learning Sentence Generation Model Development and Training

The sequence to sequence (Seq2Seq) model (Sutskever et al. 2014) was modified and trained to automatically generate the sentences based on the building-code annotations.

Model Development. The Seq2Seq model consists of two main parts: the encoder and the decoder, each consisting of several components, as illustrated in Figure 3. The encoder transforms the input (i.e., the sequence of annotated input words) into a vector representation, which captures the syntactic and semantic information of the entire input word sequence. The encoder consists of two components: the embedding layer for vectorizing the input words, and at least one LSTM layer. The decoder further transforms the vector representation generated by the encoder into the output word sequence. The decoder consists of two components: at least one LSTM layer, and the output layer for making final word prediction and thus generating the sentence.

The LSTM layers in both the encoder and the decoder aim to learn the feature representations that capture the semantic and syntactic information of the words. Each LSTM layer consists of several stacked LSTM units – each computing the feature representations based on the input information fed to the current LSTM unit, and the information propagated from the last LSTM unit so that the information from previous words could be captured. To improve the ability of the LSTM layer to deal with long-term dependencies in the building-code sentences, the bidirectional architecture was used – both the forward and backward LSTM units were considered when learning the feature representations (Huang et al. 2015). To determine the best model size for the specific training data used and the specific building-code sentence generation task, models with three different depths were tested: shallow (one LSTM layer in each of the encoder and the decoder), medium (two LSTM layers in each of the encoder and the decoder), and deep (four LSTM layers in each of the encoder and the decoder).

Model Training. Perplexity was minimized to train the deep learning model and obtain the optimal model parameters. Perplexity is defined as the inverse probability of the corpus (e.g., a group of sentences) given a language model (e.g., the Seq2Seq model), normalized by the number of words in the corpus (Jurafsky and Martin 2014).

Several training strategies were adopted. For improving computational efficiency, the training was stopped at 20 epochs or when the change of the perplexity between two consecutive training epochs was smaller than a threshold. For improving the sentence generation performance, the training practices suggested by Sutskever et al. (2014) were followed, including uniform initialization of the model parameters and gradually decreasing the model learning rate. The model was implemented in Python 3, and was run on top of Pytorch.

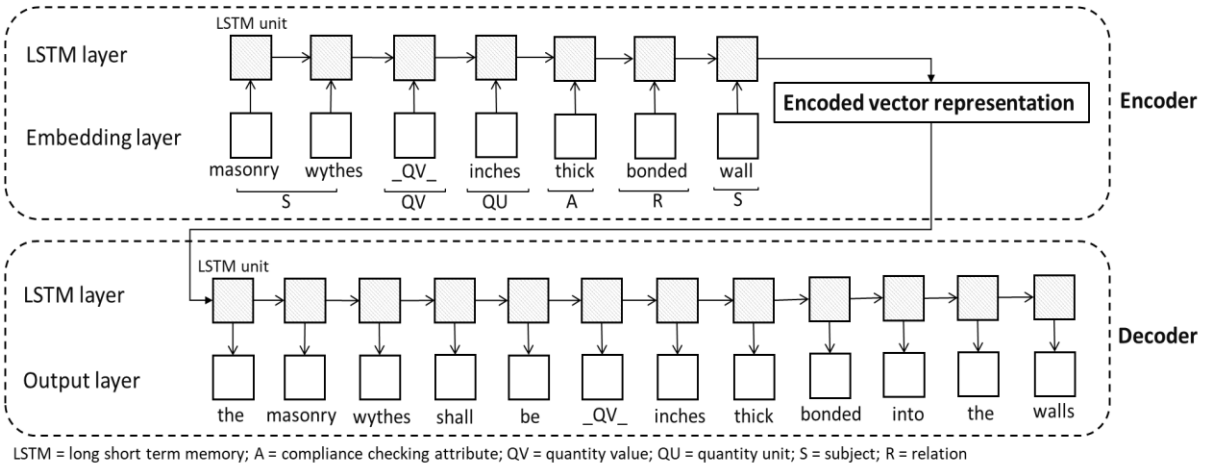

Figure 3: Sequence to Sequence (Seq2Seq) Model for Automatically Generating Building-Code Sentences

4.4 Evaluation of Generated Sentence

Two metrics were used to evaluate the comprehensibility of the generated building-code sentences: bilingual evaluation understudy (BLEU) and recall-oriented understudy for gisting evaluation (ROUGE).

BLEU is defined as the correspondence between the machine-generated text and the gold standard text (Papineni 2002), where p_n is the precision computed based on a contiguous sequence of n words, w_n is the weight corresponding to p_n, N is the longest sequence considered in calculating the metric, and c is a predefined scaling factor. $BLEU_1$ and $BLEU_2$ were used in this evaluation. A high BLEU indicates that the generated sentences correspond to the original sentences and thus the annotations used to generate these sentences are able to capture the semantics of the original sentences.

$$\text{BLEU}_N = c \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$$

ROUGE is defined as the overlap between the machine-generated text and the gold standard text (Lin 2004), where r_n is the recall computed based on a contiguous sequence of n words, w_n is the weight corresponding to r_n, and N is the longest sequence considered in calculating the metric. $ROUGE_1$ and $ROUGE_2$ were used in this evaluation. A high ROUGE indicates that the generated sentences overlap with the original sentences and thus the annotations used to generate these sentences are able to capture the semantics of the original sentences.

$$\text{ROUGE}_N = \sum_{n=1}^{N} w_n r_n$$

5. Preliminary Experimental Results

For the preliminary experiments, the size of the LSTM layers in the sentence generation model was set as 500, the dropout rate was set as 0.2, and the maximum length of the input data was set as 50. The sentence generation model optimization was performed using Adagrad, with a training data batch size of 64.

5.1 Comprehensibility with Different Types of Input Data

Two sentence generation models (of medium depth) were developed and used in the evaluation: one trained using data without preprocessing, and the other trained using preprocessed data. The evaluation results are shown in Table 2. The comprehensibility of the sentences was higher when the generation model was trained using preprocessed data, showing 4%, 3%, 3%, and 6% increase in $ROUGE_1$, $ROUGE_2$, $BLEU_1$, and $BLEU_2$, respectively. The results indicate that the two data preprocessing techniques adopted (rare-token filtering and quantity-value and reference replacement) help in generating more comprehensible building-code sentences by reducing the noise in the training data and increasing the robustness of the model.

Table 2: Comprehensibility (with medium-depth deep learning model) using different types of training data[1]

Type of training data	**ROUGE**		**BLEU**	
	$ROUGE_1$	**$ROUGE_2$**	**$BLEU_1$**	**$BLEU_2$**
Training data without data preprocessing	82%	75%	77%	67%
Training data with data preprocessing	**86%**	**78%**	**80%**	**73%**

[1]Bolded font indicates the highest performance.

5.2 Comprehensibility Using Deep-Learning Models of Different Depths

Three sentence generation models, with three different depths – shallow, medium, and deep – were developed and used in the evaluation. The evaluation results are shown in Table 3. The medium-depth model that has two LSTM layers each in the encoder and the decoder resulted in the highest comprehensibility, higher than the shallow-depth model by 6% in $ROUGE_1$, 5% in $ROUGE_2$, 3% in $BLEU_1$, and 3% in $BLEU_2$, and higher than the deep-depth model by 4% in $ROUGE_1$, 4% in $ROUGE_2$, 3% in $BLEU_1$, and 2% in $BLEU_2$. The results may indicate that the medium-depth model was most suitable for the size of the training data used – 80,000 samples, about 1,000,000 tokens. For much smaller/larger datasets or for a different type of text (e.g., different domains, different syntactic and semantic patterns, etc.), the other model depths could be used for evaluation.

Table 3: Comprehensibility with deep learning models of different depths (using preprocessed data)[1]

Deep learning building-code sentence generation model depth	**ROUGE**		**BLEU**	
	$ROUGE_1$	**$ROUGE_2$**	**$BLEU_1$**	**$BLEU_2$**
Shallow (one LSTM layer each in the encoder and the decoder)	80%	73%	77%	70%
Medium (two LSTM layers each in the encoder and the decoder)	**86%**	**78%**	**80%**	**73%**
Deep (four LSTM layers each in the encoder and the decoder)	82%	74%	77%	71%

[1]Bolded font indicates the highest performance.

5.3 Errors in Sentence Generation

Three main sources of sentence generation errors were identified based on an analysis of the experimental results. First, the building-code sentences that were used to create the training data

were automatically annotated by a pretrained machine learning model, which did not achieve 100% accuracy and thus created annotation errors. Second, the building-code sentences were collected from multiple sources including text files crawled from webpages and converted from PDF files, and noise was added during the data crawling and conversion processes. Third, the building codes contain a significant amount of non-textual data such as tables and equations, some of which are difficult to be separated from the text and thus might have contaminated the building-code sentences used to create the training data.

6. Conclusions

This paper proposed a new building-code annotation approach and a new deep learning natural language generation (NLG)-based method for evaluating annotation quality, both for supporting automated checking. The annotation approach uses requirement units that consist of semantic information elements and simple logic operators. The annotation evaluation method, which was used to evaluate the proposed building-code annotation approach, first utilizes the annotations to generate new sentences, and then evaluates the comprehensibility of the generated sentences using BLEU and ROUGE. A $ROUGE_1$ of 86%, $ROUGE_2$ of 78%, $BLEU_1$ of 80%, and $BLUE_2$ of 73% were achieved when the medium-depth sentence generation model was used, which was trained on preprocessed data. These preliminary evaluation results indicate good comprehensibility of the sentences that were generated using the proposed annotations.

This paper contributes to the body of knowledge in four primary ways. First, the paper proposed a new building-code annotation approach for supporting automated compliance checking, which uses requirement units for coverage and simplification. Second, the paper proposed a new deep learning NLG-based method for evaluating building-code annotation quality, which helps provide a well-defined method and a set of quantitative metrics for evaluation. Third, the paper leverages large-scale unlabeled data to train the deep learning models by using a pretrained domain-specific sentence annotator, which greatly reduces the manual effort needed for creating labeled data. Fourth, the experimental results show that the data preprocessing techniques and the structure of the sentence generation model could affect the comprehensibility of the generated building-code sentences and thus affect the building-code annotation evaluation.

In their future work, first, the authors plan to improve the proposed building-code annotation approach by including more complex semantic relations such as exceptions and restrictions, in order to more accurately model the relations between requirement units. Second, the authors plan to improve the proposed deep learning NLG-based annotation evaluation method by improving the quality of the training data (e.g., using rule-based or machine learning-based preprocessing methods) and testing different sentence generation model architectures. Third, and most importantly, the authors plan to integrate the proposed annotation approach with a domain ontology to help improve the performance of existing automated compliance checking systems.

Acknowledgment

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.

References

Arria NLG, (2020). Arria NLG platform, https://www.arria.com/core-tech/core-tech-overview, accessed Feb 2020.

Avitru, (2018). Spec Editor, https://avitru.com/software/spec-editor, accessed May 2018.

Digicon Information Inc., (2018). BIMdrive Specification Management Software, http://www.digicon.ab.ca/services.aspx, accessed May 2018.

Hjelseth, E. and Nisbet, N. (2011). Exploring semantic based model checking. Proc. 2010 27th CIB W78 Int. Conf., http://itc.scix.net/data/works/att/w78-2010-54.pdf, accessed Feb 2020.

Huang, Z., Xu, W. and Yu, K. (2015). Bidirectional LSTM-CRF models for sequence tagging. arXiv preprint arXiv:1508.01991.

Jurafsky, D. and Martin, J.H. (2014). Speech and language processing (Vol. 3). US: Prentice Hall.

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

Lin, C.Y. (2004). Rouge: a package for automatic evaluation of summaries. Proc. of ACL-04 Workshop. 8, pp. 74–81.

Mikolov, T., Karafiát, M., Burget, L., Černocký, J. and Khudanpur, S. (2010). Recurrent neural network based language model. Proc. of 11th Annual Conf. of ISCA. 2, pp. 1045–1048.

Pan, Y. and Zhang, L. (2020). BIM log mining: Learning and predicting design commands. Automat. Constr. 112, p.103107.

Papineni, K., Roukos, S., Ward, T. and Zhu, W.J. (2002). BLEU: a method for automatic evaluation of machine translation. Proc. 40th Annual Meeting on ACL. pp. 311–318.

Preidel, C. and Borrmann, A. (2016). Towards code compliance checking on the basis of a visual programming language. ITcon. 21(25), pp.402–421.

Ryoo, B.Y., Skibniewski, M.J. and Kwak, Y.H. (2010). Web-based construction project specification system. J. Comput. Civ. Eng. 24(2), 212–221.

Sutskever, I., Vinyals, O. and Le, Q.V. (2014). Sequence to sequence learning with neural networks. Adv. Neur. In. pp. 3104-3112.

Wen, T.H., Gasic, M., Mrksic, N., Su, P.H., Vandyke, D. and Young, S. (2015). Semantically conditioned lstm-based natural language generation for spoken dialogue systems. arXiv preprint arXiv:1508.01745.

You, Q., Jin, H., Wang, Z., Fang, C. and Luo, J. (2016). Image captioning with semantic attention. Proc. CVPR IEEE. pp. 4651–4659.

Zhang, J., and El-Gohary, N. (2013). Semantic nlp-based information extraction from construction regulatory documents for automated compliance checking. J. Comput. Civil Eng. 30(2), p.04015014.

Zhang, R., and El-Gohary, N. (2019). A machine learning-based approach for building code requirement hierarchy extraction. Proc. 2019 CSCE Annual Conf., Montreal, Canada.

Zhang, R., and El-Gohary, N. (2020). A Machine-Learning Approach for Semantically-Enriched Building-Code Sentence Generation for Automatic Semantic Analysis. Proc. 2018 ASCE CRC, Tempe, USA.