2023 European Conference on Computing in Construction
40th International CIB W78 Conference
Heraklion, Crete, Greece
July 10-12, 2023

# USING MACHINE LEARNING FOR AUTOMATED DETECTION OF AMBIGUITY IN BUILDING REQUIREMENTS

Zijing Zhang[1], Ling Ma[1]
[1]University College London, London, UK

## Abstract

The rule interpretation step is yet to be fully automated in the compliance checking process, which hinders the automation of compliance checking. Whilst existing research has developed numerous methods for automated interpretation of building requirements, none of them can identify or address ambiguous requirements. As part of interpreting ambiguous clauses automatically, this research proposed a supervised machine learning method to detect ambiguity automatically, where the best-performing model achieved recall, precision and accuracy scores of 99.0%, 71.1%, and 78.2%, respectively. This research contributes to the body of knowledge by developing a method for automated detection of ambiguity in building requirements to support automated compliance checking.

## Introduction

In the architecture, engineering and construction (AEC) industry, compliance checking is an important step where the building design is checked against requirements in building regulatory documents, recommendations and guidance (Eastman et al., 2009). Traditional compliance checking is laborious, costly, time-consuming and error-prone (Eastman et al., 2009; Macit İlal & Günaydın, 2017; Zhang et al., 2022a). To address this issue, automated compliance checking (ACC) has been a research focus in the past years. Numerous studies have proposed methods to address different aspects of the ACC challenges, including the rule interpretation and representation to a computer-readable form (Hjelseth & Nisbet, 2011; Solihin & Eastman, 2016; Yurchyshyna & Zarli, 2009; Zhang et al., 2023), the preparation of the building design model data for checking (Solihin et al., 2020), and the development of the automated compliance checking system (Kim et al., 2020; Pauwels et al., 2011).

Despite the research interest, the rule interpretation and representation step remains a bottleneck in the ACC process. Many existing methods (such as the RASE method (Hjelseth & Nisbet, 2011)) still rely on a manual or semi-automated rule interpretation, which requires extensive efforts by domain experts (Zhang & El-Gohary Nora, 2017). More recently, some automated methods, mainly based on machine learning, have been proposed and achieved satisfying performance. However, they can only deal with quantitative clauses or qualitative rules with attributes; none can deal with rules with ambiguity.

Ambiguous rules are requirements that are open to more than one interpretation. For example, the rule "Additional space may be required for special baths." is ambiguous as the additional space required is not specified and there is no explanation of what baths are special. Ambiguity is a major factor hindering fully automated compliance checking (Zhang & El-Gohary, 2022). To make matters worse, as reported by Soliman-Junior et al. (2021), up to 53% of building rules can be ambiguous, which is a considerable percentage and should be addressed.

The automation of interpreting and representing ambiguous rules would first require accurate and fast identification of ambiguous clauses in building requirements. This paper thus aims to address this issue by using machine learning methods. Identifying ambiguous clauses would be the crucial first step towards automated interpreting and representing building requirement clauses.

The remainder of this paper is structured as follows. The second section reviews related research. Then, the method used in this paper is proposed. Next, the results of this paper are presented, followed by discussions on the results. Finally, the paper offers some conclusions.

## Literature review

### Ambiguity in natural language

Ambiguity is a phenomenon in natural language that has long been studied by linguists, philosophers and psychologists. Some early studies focused on understanding ambiguity by providing classifications for different types of ambiguity. Bach (1998) identified two types of ambiguity, namely lexical (i.e., a word has more than one meaning) and structural (i.e., a phrase has more than one structure). This classification was later expanded by Berry et al. (2003) into four types: lexical, syntactical (structural), semantic and pragmatic ambiguity, where semantic ambiguity is about ambiguity related to the logic form (e.g., negation and quantifiers in predicate logic) and pragmatic ambiguity refers to more than one valid interpretation considering the context. They further provided more examples and subclasses for each type of ambiguity. For example, homonymy (e.g., bank) and polysemy (e.g., green) are two common types of lexical ambiguity. Some other studies suggested that not all

ambiguity in natural language is harmful and explored its role in communication. Piantadosi et al. (2012), for example, believed ambiguity promotes easy and effective communication when the context is informative. Larina et al. (2019) suggested that ambiguity in media discourse is a persuasion strategy and can influence public opinion.

Ambiguity has also been studied in other areas, such as the legal domain. Reidenberg et al. (2016), for example, classified vagueness in website privacy policies into four categories, including condition, generalisation, modality and numeric quantifier. Incompleteness was also identified as a type of ambiguity. Similarly, the corpus-based study on legislative texts by Li (2017) found four semantic types of vague terms, namely time, quantity, degree and category. However, these studies focused mainly on vagueness and incompleteness but neglected other aspects of ambiguity.

Ambiguity received considerable attention in the software requirement engineering (RE) domain. Extensive research has focused on the classification of ambiguity, ambiguity detection and reduction in software requirements. Kamsties & Peach (2000) developed a taxonomy of software requirements by incorporating both linguistic ambiguity and domain-specific ambiguity in requirement engineering (i.e., RE-specific ambiguity). They also proposed a checklist for easier linguistic ambiguity detection. Their later research proposed software engineering ambiguity (SE-ambiguity), which includes three categories: application domain, system domain and development domain ambiguity (Berry & Kamsties, 2004). A more recent study provided a more comprehensive ambiguity classification in the RE domain, including six categories, namely lexical ambiguity, semantic ambiguity, referential ambiguity, structural ambiguity, vagueness and incompleteness (Massey et al., 2014). While these ambiguity classifications can provide a checklist for manual ambiguity identification, some more efficient methods have been proposed, including semi-automated and automated methods. For example, Bruijn & Dekkers (2010) used the Alpino tool for automated lexical and syntactical ambiguity analysis of software requirements. Some manual tools including panel review and systematic review were also applied to improve detection accuracy. As for automated methods, natural language processing (NLP) techniques were applied. For example, Matsuoka & Lepage (2011) compared the performance of three methods, namely C-value, inverse document frequency (IDF) and similarity to WordNet in detecting ambiguity in software requirements. Similarly, Ferrari & Gnesi (2012) proposed an algorithm for identifying pragmatic ambiguity. The algorithm can first extract the sentence's "concept paths" and compare their similarities. A similarity score lower than the given threshold will lead to the sentence being regarded as having pragmatic ambiguity. Other research focused on ambiguity reduction. They either proposed a formal representation to represent natural language with less ambiguity (Osborne & MacNish, 1996) or used constrained language (Popescu et al., 2007). However, both methods have limited expressiveness and cannot fully capture the meanings of natural language.

In contrast with the software engineering domain, ambiguity in building requirements has received little attention. Only a handful of studies have mentioned ambiguity. In a study on healthcare facility requirements, Soliman-Junior et al. (2020) found ambiguity in spatial connectivity rules (e.g., adjacent to). They further suggested using semantic enrichment to address the ambiguity issues in adjacency and containment clauses. Their more recent work found that there were two types of ambiguous clauses leading to subjectivity; one is natural, while the other is artificial (Soliman-Junior et al., 2021). Natural subjectivity is clauses that include abstract elements (e.g., design flexibility) and cannot easily be made unambiguous. Artificial subjectivity, however, can be avoided if they are carefully written using clear and precise terms. In the classification of building requirements proposed by Zhang et al. (2022b), clauses concerning quality and/or aesthetics are regarded as ambiguous. Several studies attempted to address the compliance checking of ambiguous clauses. For example, Hjelseth (2013) proposed the Test Indicator Objectives (TIO) methodology to transform ambiguous phrases into quantitative metrics (e.g., well illuminated to minimum illumination in lux). Li et al. (2020) developed an automated method that used spatial artefacts (i.e., functional space, visibility space, movement space) to deal with spatial rules with ambiguity. They pointed out that the disambiguation of rules needs to be backed by related evidence. However, both methods still rely on transforming ambiguous rules into quantitative metrics.

**NLP for text classification in the AEC industry**

Natural language processing (NLP) techniques have been widely used to deal with text-related issues in the AEC industry, such as classifying documents, information extraction from documents, and ontology engineering. In this section, we will focus specifically on the methods for classifying texts in the AEC industry, mainly including contract and building requirement documents.

Text classification (TC) is a subdomain of NLP. Depending on the number of output categories and whether one sentence can belong to more than one category, TC problems can be further divided into binary TC, multi-class TC and multi-label TC. Binary TC is when there are only two possible outputs and one sentence can only belong to one category. Multi-class TC refers to sentences that belong to one of three or more mutually exclusive classes. Multi-label TC means more than one category can be assigned to the same sentences and there are more than two categories in total.

In literature, various methods have been proposed to address the three types of TC problems in the AEC industry. Some studies focused on binary TC. Hassan & Le (2020), for example, proposed a method to identify

requirements and non-requirements in construction contracts. They experimented using the word2vec model, bag of words model and bag of n-grams model using four ML algorithms, where the model built on support vector machines (SVM) achieved the best accuracy of 95%. Also dealing with contracts, Candaş & Tokdemir (2022a) implemented an ML and rule-based method to detect vagueness in the *FIDIC Silver Book*. The study revealed that a rule-based approach yielded promising performance on recall, precision and accuracy scores (89.7%) when seven vague terms were selected. The best-performing ML model achieved a lower accuracy at 80%. However, the authors pointed out that a rule-based approach is typically harder to maintain.

Other studies used various methods to address multi-class or multi-label TC. One of the first studies on the multi-class TC problem was Caldas et al. (2002), which proposed an automated method for classifying construction project documents. They evaluated the performance of classifiers based on four ML algorithms SVM, Rocchio algorithm, Naïve Bayes, and k-nearest neighbours), and the commercial software IBM Miner for Text, where SVM performed the best. To address multi-label TC problems, two commonly implemented strategies are problem transformation and algorithm adaptation. Based on the former strategy, Salama & El-Gohary (2016) developed a hybrid method based on a supervised ML algorithm and a deontic model to classify requirement clauses into scope-related categories, such as environmental or emergency management. This method proved very effective and yielded a perfect recall score and 96% precision. Candaş & Tokdemir (2022b) used a supervised ML approach with a bag of n-grams representation for multi-label classification of FIDIC contracts. They found that the model based on the SVM algorithm had the best performance, with a precision score of 0.952, yet the recall is only 0.786. A more advanced method has been developed by Moon et al. (2022) to detect different types of contractual risks. They used bidirectional encoder representations from transformers (BERT) method, which achieved 88.9% accuracy on validation and 93.4% recall on the test dataset. The BERT method showed dominant performance compared with the model built on SVM and simple neural networks. Unlike the above-mentioned research, Zhou & El-Gohary (2016b) approached the multi-label TC problem on environment-related building requirements using an ontology-based algorithm. A domain ontology on environmental requirements was developed to conceptualise related knowledge in the environmental requirements to improve the classifier performance. They reported that this ontology-based approach consistently outperformed ML-based methods in tests.

### Research gap

There have been extensive research efforts in automated compliance checking. While various methods have been proposed to manually or automatically classify, extract, interpret and represent building requirements, none can deal with ambiguous requirements. Although some scholars have shed light on the nature of ambiguity and/or subjectivity in building requirements, the categories they proposed are normally too broad and lack sufficient details and instructions for manual ambiguity detection. In addition, no automated or semi-automated methods have been proposed for detection of ambiguity in building requirements, which have the potential to greatly improve the efficiency of this step. This paper aims to propose an ML method to detect ambiguity in building requirements automatically. The proposed method is detailed in the "Methodology" section.

## Methodology

Automated ambiguity detection is essentially a text classification (TC) problem. Specifically, in this paper, as 1) there are only two possible results (i.e., ambiguous or unambiguous) for each given sentence; 2) the possible results are mutually exclusive, it is a binary TC problem. As is shown in the literature review section, the supervised machine learning method has been widely used to deal with binary TC problems (Candaş & Tokdemir, 2022a; Hassan & Le, 2020) and TC problems on building requirements (Salama & El-Gohary, 2016; Zhou & El-Gohary, 2016a), where it has demonstrated promising performance. Another widely used method for TC is the rule-based approach. Although the rule-based approach can outperform supervised machine learning in some cases, it has two main drawbacks: 1) it is time-consuming as rules are handcrafted; 2) it is less flexible as changing or expanding the dataset typically requires more rules (Schütze et al., 2008). This research thus selected the supervised machine learning method for the automated identification of ambiguous clauses.

The main steps of implementing a supervised machine learning model include model building and model evaluation, which are presented in Figure 1 and will be detailed in the following two subsections.

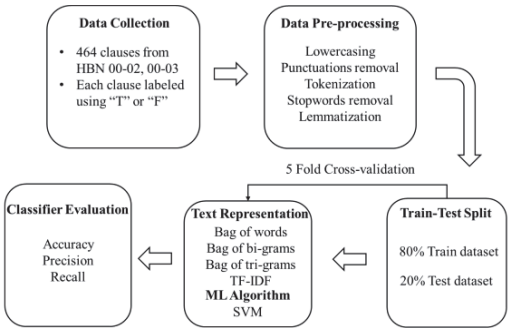


*Figure 1. The workflows of the supervised ML method*

### Building the machine learning models

To build a machine learning model, the first step is preparing the dataset. The authors selected all requirement clauses from the Health Building Notes (HBN) 00-02 and 00-03 (healthcare building requirements in England and Wales), as they have been reported to have ambiguous

clauses by Soliman-Junior et al. (2021). The data corpus includes 464 sentences, where 237 sentences are ambiguous and 227 are unambiguous. Ambiguous sentences are sentences that have multiple valid interpretations. Examples of such sentences include those with words or phrases that cause borderline cases (e.g., sufficient), lack detailed information (e.g., special baths), etc. The authors reviewed all sentences and gave a label to each sentence. Either a "T" label (denoting an ambiguous sentence) or an "F" label (meaning an unambiguous sentence) was assigned to each sentence to provide the ground truth of the classification.

The next step is data pre-processing. Five main steps were conducted. First, all texts were transformed into lowercase as ambiguity is case-insensitive. Second, the punctuation marks in the sentences were removed. This choice was made because in the authors' preliminary exploration, punctuation does not contribute much to the ambiguity of the sentence. The third pre-processing step is tokenization, where the texts are split into tokens (subunits of the sentence). Tokenization is an essential step in TC, as the ML algorithms can only analyse texts in the form of feature vectors. Next, common English stopwords (e.g., a, have) were removed by using the Natural Language Toolkit (NLTK) in Python (Bird, 2006), as stopwords mainly include non-distinctive features that are not helpful for the ML model. Then, lemmatization was performed using WordNetLemmatizer in NLTK to turn each term into its root form (e.g., rooms to room). Lemmatization helps keep the vocabulary in the data corpus small and reduces the number of features. Finally, the dataset was randomly split into a training dataset to train the machine learning model and a test set to test the model performance, where 80% of the data was in the training set and 20% was in the test set.

For the feature extraction step, three types of methods were experimented with, namely bag of words (BOW), bag of n-grams and term frequency-inverse document frequency (TF-IDF). BOW is the most widely used text representation (Schütze et al., 2008). Each sentence is represented by a vector with the counts of each word. It has the advantages of being simple and computationally efficient. However, the main drawback of the BOW representation is that it ignores the sequence of words. Bag of n-grams refers to a group of models such as unigram (words, same as BOW), bi-grams (pairs of words) and tri-grams (three words), etc., where n is an integer. It considers several consecutive words (depending on the value of "n") so that partial information on the order of words and the contexts are included (Joulin et al., 2016). Another text representation method used in this study is TF-IDF. It is a statistical method that evaluates the relevant importance of a term in a document (Lam & Lee, 1999). It does so by considering the frequency of a term in a statement and its representativeness in the whole document (i.e., the weighting will be lower for a term if it frequently appears both in a statement and the whole document) (Sebastiani, 2002).

After the feature extraction step, the machine learning algorithm support vector machines (SVM) was used to build the classification models. SVM aims to find the best decision surface for separating the training data points into classes in a high-dimensional feature space (Joachims, 1998). It can be used to classify both linear and non-linear data. It has been reported as one of the best-performing ML algorithms in previous TC studies on construction documents (Candaş & Tokdemir, 2022b; Hassan & Le, 2020).

The ML models were then trained on the training dataset and tested on the test dataset to generate performance results, which will be presented in the results and discussions section.

**Evaluating the machine learning models**

The machine learning models were evaluated using three metrics, namely accuracy, precision and recall. To calculate the three metrics, the total number of predictions needs to be first understood. Its calculation formula is shown in equation (1). TP (true positive) means the number of correct predictions on a correct classification; TN (true negative) is the number of incorrect predictions on an incorrect classification; FN (false negative) refers to the number of incorrect predictions on the correct classification and FP (false positive) means the number of correct predictions on the incorrect classification (Sebastiani, 2002). The calculation formulas for the three metrics are shown in equations (2), (3) and (4), respectively. Accuracy measures the percentage of correct predictions. A higher accuracy score denotes better model performance.

$$Total\ number\ of\ predictions = TP + TN + FP + FN \quad (1)$$

$$Accuracy = \frac{(TP+TN)}{Total\ number\ of\ predictions} \quad (2)$$

$$Precision = \frac{TP}{TP+FP} \quad (3)$$

$$Recall = \frac{TP}{TP+FN} \quad (4)$$

Despite being easy to calculate, the disadvantage of accuracy is not being discriminant and distinctive enough to select the best model (Huang & Ling, 2007). Hence, this study also used precision and recall as evaluation metrics. Precision measures the proportion of correct positive predictions. Recall, also known as sensitivity, denotes the effectiveness of the ML model on positive class identification. Hence, in general, the higher the precision and recall scores, the better the model performance.

## Results and Discussions

The authors implemented the above-mentioned ML methods using Python 3.7. Hyperparameter tuning (based

on grid search) was conducted for each model to select the best-performing parameters for model building. To alleviate potential overfitting issues (i.e., only the set of distinctive features is selected due to a small feature set), k-fold cross-validation procedures are applied. All machine learning models were able to complete the identification fully automatically within seconds. In this section, the authors look at the performance of different models based on three metrics: accuracy, precision and recall.

*Table 1: Mean performance scores of the four ML models*

| Performance score (mean) | Bag of words | Bag of bi-grams | Bag of tri-grams | TF-IDF |
|---|---|---|---|---|
| Accuracy | 82.2% | 75.5% | 78.2% | 83.0% |
| Precision | 80.3% | 73.1% | 71.1% | 82.5% |
| Recall | 88.3% | 84.7% | 99.0% | 86.2% |

The mean accuracy, precision and recall scores of different models with 5-fold cross-validation procedures and varying feature representation methods are shown in Table 1. As can be seen from Table 1, the model based on the TF-IDF feature weighting method performed the best among the four models, with an average accuracy score of 83.0%. Among the three n-grams models, the unigram model (bag of words model) yielded the best accuracy results, where the mean score was 82.2%. As the number of grams increased, the mean accuracy score first fell to 75.5% in the bi-grams model and then increased to 78.2% in the tri-gram model.

The accuracy results indicate that the increase in the size of grams is not helpful. A similar binary TC study by Hassan & Le (2020) also showed that the unigram model achieved the highest accuracy score compared with the bi-grams and tri-grams models. The data sparsity issue could be a reason for the reduction in accuracy in the bi-grams and tri-grams models (Farhoodi et al., 2011). The reason for the better performance of the tri-grams model over the bi-grams model could be that the tri-grams model helped the ML algorithm learn more semantic information in three consecutive words over two-word phrases, thereby helping the detection of ambiguity.

As for the precision and recall scores, they are typically two duelling performance metrics (Buckland & Gey, 1994). In this study, the recall score is more important than precision and accuracy because not omitting a single ambiguous sentence is more critical than 1) ensuring every ambiguous sentence detected by the ML model is actually ambiguous and/or; 2) ensuring all the predictions are correct (no matter positive or negative predictions). Ideally, a 100% recall score means domain experts can rest assured that all ambiguous sentences are among those identified by the ML model.

As shown in Table 1, the highest mean precision was achieved by the model based on the TF-IDF representation (82.5%), followed by the unigram model, which yielded a mean precision score of 80.3%. As the number of grams increased, a drop in the mean precision score was observed. The precision scores of models based on bi-grams and tri-grams were only 73.1% and 71.1%, respectively. Despite the low precision scores, the tri-grams model performed the best regarding the recall score (99.0%), while the mean recall scores of all other models were below 90%.

Overall, the satisfactory performance of the supervised ML models proposed in this paper shows that a supervised ML method is a viable and reliable way to detect ambiguity in building requirements. As the recall score is the most important metric among the three evaluation metrics, the model based on tri-grams with the SVM algorithm is the best-performing model, which achieved an almost perfect mean recall score of 99.0 % and a mean precision and accuracy score of 71.1% and 78.2%, respectively.

## Conclusions

Accurate and efficient rule interpretation is a bottleneck in the ACC process. The current automated interpretation methods are often incapable of dealing with ambiguous clauses. A quick and reliable way to identify ambiguous clauses is a crucial first step in addressing this issue. This paper proposed a supervised ML method to detect ambiguity in building requirements automatically. Four models using the bag of words, bag of bi-grams, bag of tri-grams and TF-IDF representations based on the SVM algorithm were experimented. They were trained and tested on 464 building requirements collected from Health Building Notes in England, with train and test percentages of 80% and 20%, respectively. All four models showed significant improvement in the efficiency of ambiguity detection compared with manual identification and finished detection in seconds. Among the four models, the bag of tri-grams model achieved the best recall score at 99.0%, while the TF-IDF model achieved the best accuracy score of 83.0%.

This research is the first to propose an automated method for ambiguity detection in building requirements. The proposed method significantly reduced the manual effort in detecting ambiguity in building requirements and achieved satisfactory overall performance on three evaluation metrics. The recognised ambiguous clauses can then be reviewed and interpreted by domain experts to support the automated compliance checking process. Practitioners in the AEC industry can benefit from using this tool to improve the accuracy and efficiency of checking regulations before design submission. This tool could also help improve regulators' regulations drafting to avoid excessive ambiguity.

This research has some limitations. Firstly, the dataset only includes 464 sentences. More data can be added to improve the model performance further and alleviate the data overfitting issue. Secondly, the authors only experimented with the SVM algorithm for model building. In future research, more complex ML models such as word embeddings can be used to see if the performance can be improved. Thirdly, this paper only experimented with one data pre-processing method and a 4:1 training-test split ratio. Different data pre-processing

methods and training-test set ratios can be tested in future research to achieve better performance. Future research can also widen the research scope by considering the automatic detection of more than individual sentences or detecting other valuable attributes, such as complexity, to facilitate the ACC process.

## References


Bach, K. 1998. Routledge Encyclopedia of Philosophy: Index. Taylor & Francis.

Berry, D. M. & Kamsties, E. 2004. Ambiguity in requirements specification. *Perspectives on software requirements.* Springer.

Berry, D. M., Kamsties, E. & Krieger, M. M. 2003. From contract drafting to software specification: Linguistic sources of ambiguity-a handbook. *Perspectives on Software Requirements, Series: The Springer International Series in Engineering and Computer Science* 753.

Bird, S. NLTK: the natural language toolkit. 2006. 69-72.

Bruijn, F. d. & Dekkers, H. L. Ambiguity in natural language software requirements: A case study. 2010 2010. Springer, 233-247.

Buckland, M. & Gey, F. 1994. The relationship between recall and precision. *Journal of the American society for information science* 45: 12-19.

Caldas, C. H., Soibelman, L. & Han, J. 2002. Automated classification of construction project documents. *Journal of Computing in Civil Engineering* 16: 234-243.

Candaş, A. B. & Tokdemir, O. B. 2022a. Automated Identification of Vagueness in the FIDIC Silver Book Conditions of Contract. *Journal of Construction Engineering and Management* 148: 04022007.

Candaş, A. B. & Tokdemir, O. B. 2022b. Automating Coordination Efforts for Reviewing Construction Contracts with Multilabel Text Classification. *Journal of Construction Engineering and Management* 148: 04022027.

Eastman, C., Lee, J.-m., Jeong, Y.-s. & Lee, J.-k. 2009. Automatic rule-based checking of building designs. *Automation in Construction* 18: 1011-1033.

Farhoodi, M., Yari, A. & Sayah, A. N-gram based text classification for Persian newspaper corpus. 2011. IEEE, 55-59.

Ferrari, A. & Gnesi, S. 2012. Using collective intelligence to detect pragmatic ambiguities. 2012. IEEE, 191-200.

Hassan, F. u. & Le, T. 2020. Automated requirements identification from construction contract documents using natural language processing. *Journal of Legal Affairs and Dispute Resolution in Engineering and Construction* 12: 04520009.

Hjelseth, E. 2013. Experiences on converting interpretative regulations into computable rules. *Presented at TKS* 2013: 01-15.

Hjelseth, E. & Nisbet, N. 2011. Capturing normative constraints by use of the semantic mark-up RASE methodology. 2011. 1-10.

Huang, J. & Ling, C. X. Constructing New and Better Evaluation Measures for Machine Learning. 2007 2007. 859-864.

Joachims, T. Text categorization with support vector machines: Learning with many relevant features. 1998 1998. Springer, 137-142.

Joulin, A., Grave, E., Bojanowski, P. & Mikolov, T. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Kamsties, E. & Peach, B. 2000. Taming ambiguity in natural language requirements. 2000.

Kim, I., Choi, J., Teo, E. A. L. & Sun, H. 2020. Development of K-Bim E-Submission Prototypical System for the Openbim-Based Building Permit Framework. *Journal of Civil Engineering and Management* 26: 744-756.

Lam, S. L. Y. & Lee, D. L. Feature reduction for neural network based text categorization. 1999. IEEE, 195-202.

Larina, T., Ozyumenko, V. & Ponton, D. M. 2019. Persuasion strategies in media discourse about Russia: Linguistic ambiguity and uncertainty. 15: 3-22.

Li, B., Dimyadi, J., Amor, R. & Schultz, C. 2020. Qualitative and traceable calculations for building codes. 2020. 69-84.

Li, S. 2017. A corpus-based study of vague language in legislative texts: Strategic use of vague terms. *English for Specific Purposes* 45: 98-109.

Macit İlal, S. & Günaydın, H. M. 2017. Computer representation of building codes for automated compliance checking. *Automation in Construction* 82: 43-58.

Massey, A. K., Rutledge, R. L., Antón, A. I. & Swire, P. P. 2014. Identifying and classifying ambiguity for regulatory requirements. 2014 IEEE 22nd international requirements engineering conference (RE), 2014. IEEE, 83-92.

Matsuoka, J. & Lepage, Y. Ambiguity spotting using wordnet semantic similarity in support to recommended practice for software requirements specifications. 2011 2011. IEEE, 479-484.

Moon, S., Chi, S. & Im, S.-B. 2022. Automated detection of contractual risk clauses from construction specifications using bidirectional encoder

representations from transformers (BERT). *Automation in Construction* 142: 104465.

Osborne, M. & MacNish, C. K. 1996. Processing natural language software requirement specifications. 1996. IEEE, 229-236.

Pauwels, P., Van Deursen, D., Verstraeten, R., De Roo, J., De Meyer, R., Van de Walle, R. & Van Campenhout, J. 2011. A semantic rule checking environment for building performance checking. *Automation in Construction* 20: 506-518.

Piantadosi, S. T., Tily, H. & Gibson, E. 2012. The communicative function of ambiguity in language. *Cognition* 122: 280-291.

Popescu, D., Rugaber, S., Medvidovic, N. & Berry, D. M. 2007. Reducing ambiguities in requirements specifications via automatically created object-oriented models. Monterey Workshop, 2007. Springer, 103-124.

Reidenberg, J. R., Bhatia, J., Breaux, T. D. & Norton, T. B. 2016. Ambiguity in privacy policies and the impact of regulation. *The Journal of Legal Studies* 45: S163-S190.

Salama, D. M. & El-Gohary, N. M. 2016. Semantic text classification for supporting automated compliance checking in construction. *Journal of Computing in Civil Engineering* 30: 04014106.

Schütze, H., Manning, C. D. & Raghavan, P. 2008. *Introduction to information retrieval.* Cambridge University Press Cambridge.

Sebastiani, F. 2002. Machine learning in automated text categorization. *ACM computing surveys (CSUR)* 34: 1-47.

Solihin, W., Dimyadi, J., Lee, Y.-C., Eastman, C. & Amor, R. 2020. Simplified schema queries for supporting BIM-based rule-checking applications. *Automation in Construction* 117.

Solihin, W. & Eastman, C. 2016. A knowledge representation approach in BIM rule requirement analysis using the conceptual graph. *Journal of Information Technology in Construction* 21: 370-402.

Soliman-Junior, J., Pedo, B., Tzortzopoulos, P. & Kagioglou, M. 2020. The Relationship Between Requirements Subjectivity and Semantics for Healthcare Design Support Systems. 2020. Springer, 801-809.

Soliman-Junior, J., Tzortzopoulos, P., Baldauf, J. P., Pedo, B., Kagioglou, M., Formoso, C. T. & Humphreys, J. 2021. Automated compliance checking in healthcare building design. *Automation in Construction* 129.

Yurchyshyna, A. & Zarli, A. 2009. An ontology-based approach for formalisation and semantic organisation of conformance requirements in construction. *Automation in Construction* 18: 1084-1098.

Zhang, J. & El-Gohary Nora, M. 2017. Semantic-Based Logic Representation and Reasoning for Automated Regulatory Compliance Checking. *Journal of Computing in Civil Engineering* 31: 04016037.

Zhang, R. & El-Gohary, N. 2022. Natural language generation and deep learning for intelligent building codes. *Advanced Engineering Informatics* 52: 101557.

Zhang, Z., Ma, L. & Broyd, T. 2022. Towards fully-automated code compliance checking of building regulations: challenges for rule interpretation and representation. 2022a. EC³ (European Conference on Computing in Construction).

Zhang, Z., Nisbet, N., Ma, L. & Broyd, T. 2022. A multi-representation method of building rules for automatic code compliance checking. European Conference on Product and Process Modeling 2022, 2022b Trondheim, Norway.

Zhang, Z., Nisbet, N., Ma, L. & Broyd, T. 2023. Capabilities of rule representations for automated compliance checking in healthcare buildings. *Automation in Construction* 146: 104688.

Zhou, P. & El-Gohary, N. 2016a. Domain-Specific Hierarchical Text Classification for Supporting Automated Environmental Compliance Checking. *Journal of Computing in Civil Engineering* 30.

Zhou, P. & El-Gohary, N. 2016b. Ontology-Based Multilabel Text Classification of Construction Regulatory Documents. *Journal of Computing in Civil Engineering* 30.