Regulatory Information Transformation Ruleset Expansion to Support Automated Building Code Compliance Checking

Xiaorui Xue, S.M.ASCE [1]; Jiansong Zhang, Ph.D., A.M.ASCE [2]

Abstract

The traditional building code compliance checking process mainly relies on design reviewers to review design documents or models manually. The intensive manual effort needed makes this process time-consuming, costly, and error-prone. Automated compliance checking (ACC) could be a promising upgrade of the traditional manual code compliance checking. With a reduced workload on design reviewers, ACC is cheaper, faster, and immune to human errors. To support ACC, building code requirements need to be represented in a computer-processable format to enable automated reasoning, which will in turn allow an automated assessment of the building design's compliance status with building codes. A major limitation of many existing ACC systems/methods is their limited range of checkable building code requirements. To address that, the state of the art uses pattern matching-based rules to transform building code requirements into computable formats automatically, but the ruleset was developed and tested only on a few chapters of building code requirements. An efficient ruleset expansion method is needed to increase its range of checkable building code requirements at a low cost to bring ACC systems closer to full deployment. In this paper, the authors proposed a new regulatory information transformation ruleset expansion method for expanding an existing ruleset. This method can expand the range of checkable code requirements of ACC systems without significant manual effort. The proposed ruleset expansion method takes an iterative approach to ensure the generality and validity of new pattern matching-based rules and the quality of information transformation results. The expanded ruleset was tested on generating logic clauses from Chapter 5 of the International Building Code 2015. Compared to the baseline ruleset, the expanded ruleset increased the predicate-level precision, recall, and F1-score of the logic clause generation by 10.44%, 25.72%, and 18.02%, to 95.17%, 96.60%, and 95.88%, respectively.

1. Introduction

A broad range of building codes govern the architecture, engineering, and construction (AEC) industry. There are building codes that are applicable to an entire building, such as the International Building Code (IBC) [1] and the International Fire Code (IFC) [2], or building codes that are applicable to a part/component of a building, such as the American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE) Standard 90.1 and the International Energy

Conservation Code (IECC). The number of existing building codes is large. For example, the State of Indiana enforces over 50 different construction-related codes, standards and other regulations [3]. The large amount of existing building codes makes traditional manual code compliance checking suffer from three major limitations: (1) a long review process, (2) high review cost, and (3) tendency to have human errors and omissions in review results [4-6]. A traditional code compliance checking process involves multiple review cycles and interactions between design reviewers and building designers. In each design review cycle, building designers submit their designs to design reviewers. If the designs have omissions or mistakes, design reviewers will return them to the building designers and ask for additional information or revisions. The interaction between building designers and design reviewers continues until the design reviewers are confident that there is no omission or mistake in the design anymore, at which point a building permit will be issued. Traditional code compliance checking is a long and expensive process. For example, the average turnaround time for issuing a building permit is 73 days and the minimum cost is 302 dollars in Chicago [7,8].

A potential solution to address the limitations of traditional manual code compliance checking is automated compliance checking (ACC). In ACC, software and algorithms reduce the workload of design reviewers by automatically checking building designs against building codes. By making the job of design reviewers easier, ACC is expected to cost less money, consume less time, and generate fewer errors than the traditional code compliance checking approach [9]. The potential benefits of ACC also include: (1) facilitating the adoption of building information modeling (BIM) in the AEC industry [10,11], (2) promoting a common information exchange format in the AEC industry [12], and (3) encouraging the use of computing techniques such as natural language processing (NLP) and logic reasoning in the AEC industry [13]. Given all the benefits, the exploration of ACC dates back to the 1960s, when Fenves [14] developed a decision table system to check the design of steel structures against the American Institute of Steel Construction (AISC) specifications. The success of Fenves paved the road to more complex ACC systems. For example, Garrett and Fenves [15] proposed the use of information networks to represent design standards in the ACC systems. At the same time, the boom of expert systems gave birth to expert ACC systems, such as the Standard Interface for Computer-Aided Design (SICAD) [16], the Standards Processing Expert (SPEX) [17], and the Design Prototype system [18].

Prior attempts at expert ACC systems ceased in the 1990s due to their high maintenance cost, but more complex expert ACC systems, such as the BCAider and the DesignCheck [19], emerged in the early 2000s [20]. In addition to expert ACC systems that aim to check general building codes, some expert ACC systems focus on building codes in a specific domain, such as (1) the Fire-Code Analyzer that checks the compliance of fire protection codes in New Zealand [17], (2) the Life Safety Code Advisor that checks against the National Fire Protection Association (NFPA) safety code in the United States, and (3) the TALLEX that checks compliance of tall buildings in the United Arab Emirates (UAE) [21]. The introduction of BIM as a digital representation of buildings led to the advancement of BIM-based ACC systems. Solibri Model Checker (SMC) started as a BIM validation tool, but automated code compliance checking plugins soon evolved [22]. The

DesignCheck checks BIM of a building against accessibility regulations [19]. The Construction and Real Estate Network (CORENET) project by the Singapore government checks BIM, instead of paper-based design documents, against barrier-free access codes and fire codes. The KBimCode can check BIM against South Korea's building codes [23].

Despite all the development and advancement of ACC systems, existing ACC systems still rely on domain experts to extract building code requirements and formalize them into a computer-processable format [24,25], such as decision tables [9], knowledge models [26], or structured rulesets [27]. The high cost and long duration involved in this manual process prevent a comprehensive testing of the regulatory information coverage of ACC systems. Many existing ACC systems only focus on a specific range of building code provisions for this reason. For example, Nguyen [28] proposed an ACC computing framework by manually converting two chapters of the International Building Code (IBC) to a hierarchical data structure. Nawari [29] manually translated a portion of the National Green Building Code Standard (ICC 700-2008) to the extensible markup language (XML) format. These efforts in academia successfully proved the concepts of many promising ACC systems, but their limited range of checkable building code requirements prevented their full-fledged industry adoption. The large amount of manual effort required to convert building codes in natural language to a computer-processable structured format hindered the full deployment of ACC systems in the AEC industry. Vendors of commercial ACC systems believed that they could convert enough building code requirements and achieve automated code compliance checking by hiring domain experts. However, the high cost of manual conversion led to this idea tested uneconomic, and most related attempts ceased in the 1990s.

To enhance the practicality of ACC systems, the extraction and transformation of building code requirements from a wider range of sources need to be automated. Fully automated processing of a wide range of building code requirements is a key functionality of envisioned future ACC systems. The authors proposed a ruleset expansion method to expand the range of checkable building code requirements of ACC systems that use a pattern matching-based regulatory information transformation ruleset to automatically transform building codes from natural language to computable logic clauses that support automated reasoning. This type of system [30], although fully-automated, has a limited range of checkable building code requirements in its current implementations. The existing automated building code transformation method [31] was established based on the assumption that a set of common patterns exist that could be used to analyze the various information elements encoded in building code requirements and transform them to an unambiguous and computable representation. It was not clear, however, how large this set of common patterns should be. As a result, based on this assumption, although theoretically there exists a "superset" of common patterns, in practice, it is likely that we will need to adopt a data-driven approach to accumulatively grow an existing set of common patterns to asymptotically approach that theoretical "superset." In line with that, the existing set of common patterns (or ruleset of pattern matching-based rules) needs to be continuously expanded to cover otherwise missed regulatory information when they are applied to unfamiliar building codes. More pattern matching-based rules are needed to bring them to practice in the AEC industry.

In this paper, the authors proposed a ruleset expansion method to expand the range of building code requirements that ACC systems can extract and transform in an efficient manner. The ruleset expansion method needs to control two aspects: (1) the validity of the ruleset, and (2) the generality of the ruleset. The proposed method therefore takes an iterative approach to pursue a balanced validity and generality of added rules while maintaining the compatibility of added rules with existing rules. The proposed method is designed to support ACC systems that check 3D digital BIM models of building designs. The design information extraction module of the ACC systems extracts building design information through a series of information extraction and transformation algorithms [32]. Then, the extracted building design information and automatically generated logic clauses are sent to the compliance checking reasoning module to generate compliance checking results [25,33,34]. Methods that extract building design information from BIM or IFC files have been well developed [35,36]. While the proposed method in this paper focuses on producing logic clauses that work with BIMs, it can potentially be used with building plans and drawings as well. The extraction of building design information from building plans and drawings could be a promising direction for future research.

2. Background

2.1 Natural Language Processing

Chowdhury defines natural language processing (NLP) as "an area of research and application that explores how computers can be used to understand and manipulate natural language text or speech to do useful things [37]." NLP includes a wide range of tasks, such as (1) information retrieval [38], (2) information extraction [39], (3) text classification [40], (4) text generation [41], (5) text summarization [42], (6) question answering [43], (7) machine translation [44], and (8) speech recognition [45]. There are two main approaches to accomplishing NLP tasks: the rule-based approach and the machine learning-based approach [46]. Rule-based NLP systems may require manual effort in rule generation, but usually outperform machine learning-based NLP systems in a specific task or in a specific domain [47]. Machine learning-based NLP systems can be further classified into "shallow" learning systems and "deep" learning systems based on the types of machine learning models they use. "Shallow" learning systems use traditional machine learning algorithms, such as support vector machines (SVMs) or decision trees, and require manual feature engineering. Deep learning systems use neural networks and do not require manual feature engineering [48]. There is no lack of efforts to use NLP in the AEC domain. For example, Tixier et al. [49] used NLP to extract the reasons for accidents from construction injury reports. Lin et al. [50] used NLP technologies to extract information from BIM. ACC research also uses NLP techniques to process building codes, for matching between concepts in building codes and concepts in BIM [51], and for converting building codes to logic clauses that support automated reasoning [52].

2.2 Part-of-Speech

Part-of-speech (POS) of a word represents its lexical and syntactic function in a sentence [43]. English words have eight basic POS categories: (1) noun, (2) verb, (3) adjective, (4) adverb, (5) pronoun, (6) preposition, (7) conjunction, and (8) interjection [54]. The same word may have different POS categories in different contexts. For example, the word "run" could be a verb in its simple present tense or past perfect tense depending on the context. In NLP systems, words are categorized into more specific POS categories to represent text more informatively. For example, the Penn Treebank Corpus classifies words into 36 POS categories [55] and the Brown corpus has 179 POS categories [56]. In the development of the Penn Treebank Corpus and the Brown Corpus above, human annotators manually assigned words to different POS categories according to their understanding of the English language and the contexts of the words. POS tagging software, which is commonly called "POS taggers," could replace annotators' manual effort in this task. POS taggers automatically determine the POS category of a word using its contextual information in an algorithmic manner [57]. POS taggers began with rule-based taggers that used a set of rules to determine the POS categories of words. These rules can be compiled by experts [58] or extracted from text algorithmically [53]. With the development and integration of machine learning, POS taggers shifted to the use of statistical models. For example, Giménez and Marquez [59] used one SVMs model to determine POS categories of known words and another SVMs model to predict those of unknown words. Brants [60] developed a POS tagger which uses Hidden Markov Models (HMM) to capture dependencies among words and determine the POS categories of words by their inter-dependencies. Plank et al. [61] proposed the use of bi-directional neural networks to accomplish multilingual POS tagging. POS tagging is an important early step of many NLP systems [59].

2.3 Ontology

Ontology is the explicit and formal description of knowledge through relationships among concepts in a domain [62]. In 1999, the World Wide Web Consortium (W3C) first developed the Resource Description Framework (RDF) language for ontology [63]. Then, it collaborated with the Defense Advanced Research Projects Agency (DARPA) to extend the RDF into a more expressive DARPA Agent Markup Language (DAML) [64] [65]. After that, many ontologies emerged, either for a specific domain (e.g., medical) [66] or for general-purpose [67]. Ontology is used to: (1) analyze and reuse domain knowledge, (2) share structured domain knowledge among people and software, (3) specify domain assumptions, and (4) distinguish domain knowledge from operational knowledge [68].

2.4 Text Similarity Measurements

Text similarity is an important benchmark in NLP. There are many ways to measure the similarity between text strings [69]. Text similarity can be measured by comparing words or characters in text strings. For example, the Levenshtein Distance [70] [71] measures the minimum number of single character transformations needed to convert one string to another. In Levenshtein Distance, 0 means two strings are identical, and the larger it is, the less similar the two strings are, with no strict upper bound. The Jaccard Distance, on the other hand, measures the number of items shared by two sets [72]. In the Jaccard distance, 0 means two sets are identical, and 1 means two sets share no common items. The Jaro Winkler Similarity [73] is an extension of the Levenshtein Distance. By normalizing the Levenshtein Distance with the length of the text string, the Jaro Winkler Similarity ranges from 0 to 1. In the Jaro Winkler Similarity, 0 means two text strings are completely different and 1 means two text strings are the same.

The inability to measure similarity between word meanings is one limitation of measuring text similarity at the word and character levels. One potential solution to the problem is representing words (and their contexts) as vectors in high-dimensional spaces. Popular text vectorization techniques include, for example, Word2Vec [74], FastText [75], and Glove [76]. The distance between meanings of the two words and their contexts can be measured by the cosine distance between the two vectors.

3. Methodology

The proposed method expands the range of processable building code requirements by adding new pattern matching-based rules to an existing ruleset. The pattern matching-based rules capture regulatory information in building codes and convert the captured information to logic clauses. The pattern matching-based rules consider both syntactic information, which is provided by POS tags (e.g., the word "height" in the phrase "building height" is a noun because it has a POS tag "NN"), and sematic information, which is provided by an ontology (e.g., the phrase "less than" after an attribute and before a value means that the attribute value must be smaller than the specified value, and the phrase "minimum clearance" means the attribute "clearance" must be greater than or equal to a specified value). For example, the pattern "subject, conjunction, subject" can be used to extract the regulatory information of which two subjects are in equivalent status. The conjunction (i.e., and, or) is a label by the POS tagger. Subjects, which were labeled as noun by the POS tagger, were further labeled as subjects after feature enhancement by the ontology. The pattern matching-based rules regulate how this method extracts building code requirements and converts them to logic causes. The logic clauses represent building code requirements in a strict horn clause (HC) format in B-Prolog syntax to avoid ambiguity in natural language and facilitate automated reasoning by the logic reasoner.

Each logic clause has a left-hand side and a right-hand side, separated by the delimiter ":-". The

left-hand side is the head of the logic clause that represents the subject of compliance checking in the logic clause, i.e., the building design component that this code requirement governs. The subject of compliance can be an entire building, a component of a building, a certain attribute of a building, or an attribute of a building component. The predicates on the right-hand side of the delimiter ":-" (i.e., in the body of the logic clause) are conditions that the subject of compliance need to meet to comply with the building code requirement. This logic clause indicates that if all predicates on the right-hand side of the delimiter ":-" are evaluated to True, then the predicate on the left-hand side of the delimiter ":-" will also be evaluated to True. In the context of this study, it means that if the subject of compliance meets all the conditions of the corresponding building code requirement, it is then considered to be compliant with the building code requirement. In the logic clauses, the conjunction relation (i.e., AND) is represented as a comma "," and the disjunction relation (i.e., OR) is represented as a semicolon ";".

One manually generated logic clause example as part of the gold standard (see details in Section 4.2 - Gold Standard Generation) is provided in Figure 1. The "*Travel_distance*" in Figure 1 is the subject of compliance and the predicates on the right-hand side describe the building code requirement that the "*Travel_distance*" need to comply with. Each predicate describes one condition that the subject needs to satisfy to comply with in the building code requirement described in the logic clause. If all of the predicates on the right-hand side are evaluated to true, the ACC system will then determine the building design to be in compliance with the corresponding building code requirement. More specifically, the predicates "*from(Travel_distance, Accessible_space), to(Travel_distance, Area_of_refugee)*" describe that the travel distance is measured from the accessible space to the refugee area. The predicate "*in_accordance_with(Travel_distance_2, section_1017_1)*" describes that the travel distance is specified in Section 1017.1 of the IBC 2015. The predicates "not *greater_than(Travel_distance, Travel_distance_2)*" require the travel distance from the accessible space to the area of refugee to be no greater than the travel distance specified in Section 1017.1 of the IBC 2015. Other predicates in the logic clause are required by the strict HC format in B-Prolog syntax for this logic rule to execute. Overall, this logic clause describes the building code requirement that the maximum travel distance from an accessible space to an area of refugee should not be greater than the travel distance specified in Section 1017.1 of the IBC 2015.

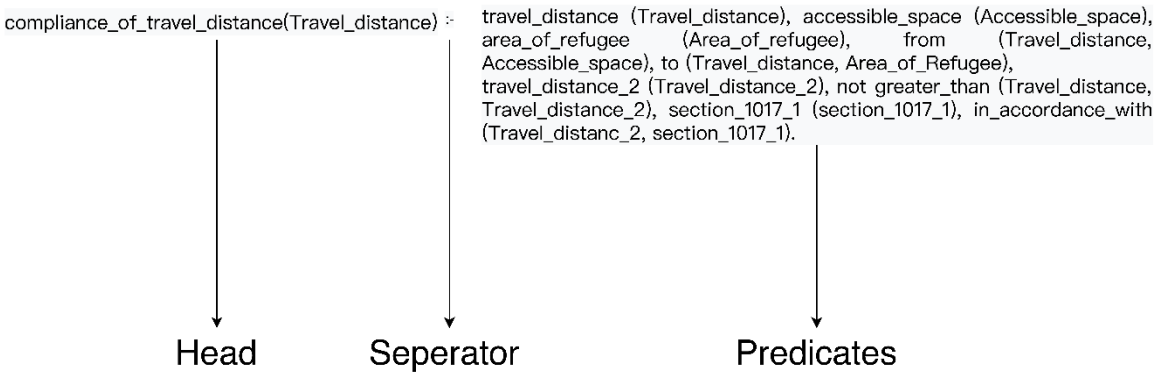


Figure 1. Example Logic Clause.

In the manual transformation of building code, domain experts complete the transformation based on their understanding of building code requirements. In our automated transformation, a pattern matching-based regulatory information transformation ruleset is used to complete this transformation automatically. To support the matching pattern development in the ruleset, building codes undergo feature enhancement by POS tagging and ontology matching (Figure 2).

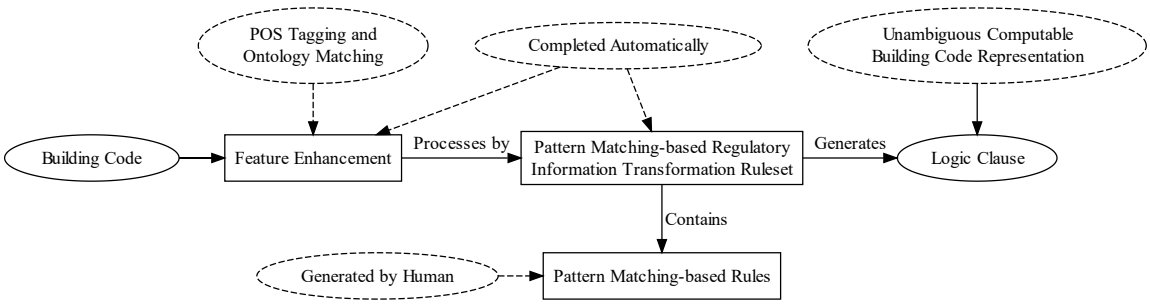


Figure 2. Automated Logic Clause Generation.

The goal of this research is to develop an efficient and effective method to extend an existing pattern matching-based regulatory information transformation ruleset. Although it is possible to develop a new ruleset from scratch, the authors chose to expand an existing ruleset developed by Zhang and El-Gohary [31], based on the assumption that asymptotic full coverage of building codes could be achieved by expanding an existing ruleset. In addition, expanding an existing ruleset, instead of generating new rulesets, has the benefits of: (1) reducing rule generation workload, and (2) allowing the expanded ruleset to capture patterns absent in the training dataset, while maintaining the compatibility of the expanded ruleset with the automated building code compliance checking system. The expansion of an existing ruleset requires new rules to be added. The added rules must meet certain standards or have certain characteristics to realize the benefit goals above. For example, the amount of effort/time spent on new rules development should be significantly less than (e.g., no greater than 20% of) the development of the original ruleset. To achieve the first goal, the number of added rules should be small. To achieve the second goal, the process of adding new rules needs to be selective. The added rules should be valid and general. A rule is valid when it correctly extracts the regulatory information it is designed to extract. A rule is considered general when it has been applied at least twice in the training dataset. The combination of multiple valid and simple pattern matching-based rules can be used to represent more complex patterns in building codes. The added rules also need to be general to capture patterns that are not in the training data. Building codes are legal documents composed by a panel of experts following strict guidelines. Therefore, the same patterns may be shared by different chapters of the building code. The generality requirement allows pattern matching-based rules to capture common patterns shared by different chapters of building codes, or different building codes. Different building codes, or at least different chapters in one building code, should follow a set of common patterns, according to English grammar and the way building codes were compiled. The ruleset expansion method proposed in this research is designed

to ensure the generality and validity of added rules. The transformation rule generation and validation are manually conducted in the proposed method. However, once the transformation rules are generated and validated, they can be used to automatically transform building code requirements into logic clauses.

3.1 Ruleset Expansion Method

The proposed ruleset expansion method takes an iterative approach to add new rules into an existing regulatory information transformation ruleset. The goal of this research is to develop an efficient method to expand an existing pattern matching-based regulatory information transformation ruleset and ensure the generality and expandability of the added rules. To achieve this goal, the authors should identify missing regulatory information and generate new rules to capture it. There are two approaches to completing this task: (1) identify all missed regulatory information and generate corresponding rules in a single pass, or (2) identify one piece of missed regulatory information at a time, generate a rule to capture the missed information, review the performance of the new rule, modify the new rule, and then proceed to the next iteration of identifying missed information. Both approaches can generate a ruleset that captures all regulatory information. However, the first approach does not consider the validity and generality of the added rules and the interaction among them (i.e., multiple rules may match the same regulatory information). The second approach iteratively adds new rules and tests them before they are eventually added to the ruleset. The second approach has the potential to generate more valid and general rules than the first approach. What really distinguishes the first and second approaches is the granularity of pattern matching-based rules. The first approach aims to extract regulatory information at the chapter level or even at the whole building code level. Whereas the second approach extracts regulatory information at the sentence level. Because logic clauses are generated at the sentence level, the second approach naturally fits better. In addition, the shorter the pattern lengths are, the more flexible they become and the better scalability the whole ruleset will have. Patterns may match the whole sentence of a building code requirement or (most likely) part of a sentence. Data-driven expansion of the existing ruleset is also made possible through the second approach, whereas it would not have been possible with the first approach. Previous rule-based NLP applications [58,77,78] with manually developed rules also supported this point. Testing rules one-by-one before they are added to the ruleset is also a more rigorous rule development process than generating all the rules and testing them together.

Because of the above-mentioned advantages, the ruleset expansion method proposed in this research takes the second approach, which is shown in Figure 3. One candidate pattern matching-based rule is generated to capture missing regulatory information one piece at a time, until all the missed regulatory information in the training dataset is captured. A version of logic clauses is first generated by the ruleset to identify missing regulatory information. If an instance of missing regulatory information is identified in this version of logic clauses, a candidate pattern matching-based rule is generated to extract it. The candidate pattern matching-based rule will be added to the ruleset if it is proven to be general and valid. The generality and validity of the candidate rule are tested by

inspecting a new version of logic clauses generated by the ruleset when the candidate rule is included. A valid rule must correctly extract the information it is designed to extract and does not introduce errors when it is applied to other parts of the training text. A general rule needs to be able to be applied at least twice in the training dataset, which means it should be applied at least once outside of the sentence it is extracted from. The validity of a rule has higher priority than its generality. If the rule introduces any errors in the training text, it will be modified until it introduces no errors. At the same time, the validity of rules will not be sacrificed to make a new rule general. It is possible that a pattern appears only once in the training text, and it is still necessary to capture an instance of regulatory information. The ruleset expansion process continues until the expanded ruleset captures all the regulatory information in the training data.

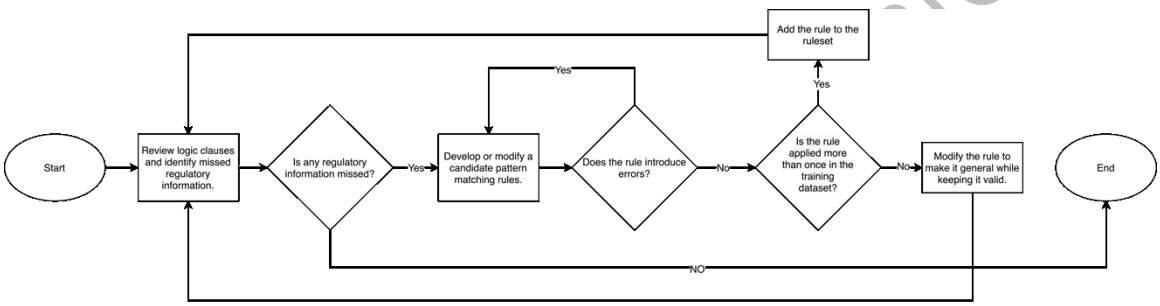


Figure 3. Ruleset Expansion Method.

3.2 Feature Enhancement

The building codes are first enhanced by POS tagging and ontology matching, to generate extra features. The enhanced building codes are more informative and support more complex operations than the original building codes without extra features. Building codes are labeled with information tags in this step. Extra features can provide more information about building code expression patterns and, therefore, increase the performance of pattern matching-based rules. To understand building codes, it requires knowledge both of the English language and of the AEC domain. The feature enhancement makes the ACC system better at processing building codes by introducing such knowledge to the ACC system. The authors apply POS tagging to generate syntactic features (i.e., for knowledge of the English language) and apply ontology matching to introduce AEC domain knowledge in this step.

The ACC system uses POS tagging, which captures the grammatical roles of words in a sentence, to generate syntactic features from building code. The same word in different POS categories can have distinct meanings. For example, when the word "run" is a verb, it means an action of moving through a space. When the word "run" is a noun, it refers to a physical object. Therefore, syntactic features together with semantic features can disambiguate words in building codes. For example, when the word "runs" is a noun in the sentence "The extensions of handrails shall be in the same direction of the flights of stairs at stairways and the ramp runs at ramps" [1] (Section 1014.6 of IBC

2015), it represents a physical object with attributes governed by the building code. However, when the word "runs" is a verb in the sentence "Where a partition containing piping runs parallel to the floor joists" [1] (Section 2308.5.8 of IBC 2015), such a possibility can be ruled out. In this research, the authors used the A Nearly-New Information Extraction System (ANNIE) POS tagger in the General Architecture for Text Engineering (GATE) [80] with proven performance in tagging building codes to generate accurate syntactic features. Such external POS taggers, which were trained on a larger body of text and fine-tuned by domain experts, can bring additional grammatical knowledge to the ACC system.

The ACC system also uses an ontology to introduce AEC domain knowledge for logic clause generation. In manual code compliance checking, reviewers already have domain knowledge needed to understand building codes, based on their education, training, and experience. However, in ACC systems, such knowledge needs to be explicitly provided. An ontology allows the ACC system to access domain knowledge and consider domain knowledge in rule generation. For example, with an ontology, the method can treat "International Fire Code" and "automatic sprinkler system" as integral phrases instead of multiple individual words. It also makes the method treat "inches" and "feet" as units specifying a numerical constraint, instead of regular nouns. In addition, the ontology also supports the disambiguation of vague terms.

The used ontology has two main types of items: (1) essential information, and (2) secondary information. Essential information includes: (1) subject of a particular regulatory requirement (e.g., building), (2) attribute (e.g., building height), (3) comparative relationship (less than, greater than), (4) quantity (e.g., value or range of value), (5) quantity unit (e.g., inch, feet), and (6) reference to other quantity. Secondary information includes restrictions and exceptions. Restrictions mean constraints to subjects and attributes [26] [31]. For example, in the sentence "Exterior exit stairways and ramps serving as an element of a required means of egress shall be open on not less than one side, except for required structural columns, beams, handrails and guards [1]," "serving as an element of a required means of egress" is a constraint to "Exterior exit stairways and ramps." Exceptions are the conditions where a requirement does not apply. The ontology was created and tested in a previous study [31]. With an expansion for this specific task, its comprehensiveness is ensured in the context of this application by enumerating all covered concepts in the corresponding code requirements. The ontology is also scalable. Similar to the ruleset itself, the ontology could also be accumulatively and continuously developed to fulfill the need for processing different building codes, until it reaches or asymptotically approaches the saturated state where any potentially related concept to building codes is included. It is editable in GATE [80] or using a plain text editor. Ontology editing tools provide support for the scalability of the ontology as the size and complexity of the ontology increases.

3.3 Pattern Extraction

There are two approaches to extracting regulatory information from building codes: the top-down approach, and the bottom-up approach. In the top-down approach, the information extraction

algorithm constructs a global logic clause framework that matches the overall structure of a sentence and fills in the building code requirements into the framework. In the bottom-up approach, the information extraction algorithm captures building code requirements at a local level and assembles them into a logic clause. The building codes consist of a lot of long and complex sentences with diverse structures. The versatility and complexity of building code sentences means developing enough sentence-level frameworks to accommodate all sentences in building codes may require a similar amount of manual effort as directly converting building codes to logic clauses. It is possible that every sentence, or at least every few sentences, requires a different framework. The authors propose the use of the bottom-up approach. The complex and versatile structures of building code sentences may require many complex sentence-level frameworks, but pattern matching-based rules can assemble these structures from simple local patterns.

4. Experiment

4.1 Ruleset Expansion Experiment

The authors tested the effectiveness of the proposed ruleset expansion method by measuring its precision, recall, and F1-score in logic clause generation, which in turn tested the generality of the original ruleset. Chapter 10 of the International Building Code 2015 (IBC 2015) was selected as the training data for the experiment and Chapter 5 of the IBC 2015 was selected as the testing data. Zhang and El-Gohary developed the original ruleset based on Chapters 12 and 23 of IBC 2006 [31]. The authors used the ruleset expansion method to generate new rules based on Chapter 10 of the IBC 2015 and tested the expanded ruleset on Chapter 5 of the IBC 2015, in comparison with the original ruleset.

In the first part of the experiment, the original ruleset generated a baseline version of logic clauses from the training data. The training data was first pre-processed by a POS tagger and an ontology to generate enhanced features. The POS tagger used in this research is the ANNIE tagger from the GATE tool [79]. The authors used the ontology developed in [31] with expansions on Chapters 5 and 10 of the International Building Code 2015. After that, the authors used the ruleset expansion method to expand the original ruleset.

First, the authors identified missing regulatory information and updated the original ruleset with a candidate pattern matching-based rule to capture the missing regulatory information. For example, the original ruleset did not have enough patterns to extract all the essential information for requirements that are described using negation together with a past participle verb, so a corresponding pattern and candidate rule was added. The expanded ruleset also includes all rules in the original ruleset. The authors added 64 new rules to the original ruleset, a much smaller number compared to the 306 rules already in the original ruleset. Two of the 64 new rules were developed to extract missed regulatory information in the example type mentioned above. One rule with the pattern "modal verb, negation, base form verb, [adjective, past participle verb, past tense verb]" was

generated to extract the regulatory requirement of a subject. This rule can process building code requirement sentences like "A basement (candidate subject) provided with one exit shall (modal verb) not (negation) be (base form verb) located (past participle verb) more than one story below grade plane" (Section 1006.3.2.2 of IBC 2015) [1], and "The area of a Group F-2 or S-2 building (candidate subject) no more than one story in height shall (modal verb) not (negation) be (base form verb) limited (past participle verb) where the building is surrounded and adjoined by public ways or yards not less than 60 feet in width" (Section 507.3 of IBC 2015) [1]. The first subject is extracted by identifying the first subject candidate to the left of (not necessarily immediately next to) the relationship. This arrangement makes pattern matching flexible. The rule is both general and valid, because it was applied twice in the training dataset and correctly extracted the regulatory information it was designed to extract. Another pattern "candidate subject, preposition, comparative relation, value, unit" was generated to extract the quantitative regulatory requirement of a subject. This rule can process building code requirement sentences like, "The ladder or steps shall not encroach into the required dimensions of the window well (candidate subject) by (preposition) more than (comparative relation) 6 (value) inches (unit)." (Section 1030.5.2 of IBC 2015) [1]. Only the "of" relationship between "the required dimension" and "the window well" was extracted by the original ruleset. The newly added rule was not general in the current training dataset (i.e., it was applied only once in the training dataset), but it was still valid, because it correctly extracted the regulatory information it was designed to extract. Although the rule was not general, it was still needed to capture an instance of regulatory information. Therefore, it was still added to the ruleset. The complete set of the 64 rules can be found in Appendix A. In the second part of the experiment, the expanded ruleset was tested on Chapter 5 of the IBC 2015 to automatically convert building codes to logic clauses. The automatically generated logical clauses were compared against a gold standard.

4.2 Gold Standard Generation

Chapters 10 and 5 of the IBC 2015 were transformed into logic clauses by three annotators semi-automatically to create a gold standard of information transformation and logic clause generation. All three annotators have background AEC knowledge to understand building codes, and necessary skills to transform building codes into logic clauses. The authors provided the annotators a clear annotation protocol, a brief training section before annotation, and machine-generated logic clauses as reference during their annotation. They worked independently without access to the logic clauses generated by other annotators. However, they were presented with the machine-generated logic clauses, which could help annotators align with the rule generation mechanism of pattern matching-based rules, achieve higher inter-annotator agreement, and reduce rule generation time. It also ensures the compatibility of human-generated logic clauses with the automated code compliance checking system.

Annotators were required to use the exact words that came from the building code in their generated logic clauses. For example, if the building code uses the word "exterior" for exterior walls,

annotators must also use the word "exterior" in their generated logic clauses to represent exterior wall, rather than using "external wall" or other names. Therefore, the problem that annotators may use different words for the same meaning is prevented. The product of the manual transformation process was three versions of logic clauses, with each version independently and manually generated by one of the annotators. After that, annotators reviewed each other's work and collectively generated the final gold standard. All annotators agreed that the gold standard represents the meaning of building codes accurately and approved it.

To evaluate the quality of human-generated logic clauses, the authors measured the similarity between the logic clauses generated by different annotators. Because annotators transformed the same building code, they should generate similar logic clauses. A high similarity among human-generated logic clauses of different annotators indicates a high quality of the logic clause generation. The authors chose to measure logic clause similarity by comparing characters and words at the string level in this study. While text vectorization and cosine similarity measure the meaning-wise similarity of natural language text, because the logic clauses generated in this study are not in natural language and our similarity measurement focuses on the existence of logic clause components rather than the meaning of the logic clauses, vectorization of text and cosine similarity were therefore not used. The authors measured the similarity among logic clauses in two different ways: (1) the Levenshtein Distance and the Jaro Winkler Similarity were used to measure the character-level similarity between the human-generated logic clauses; and (2) the Jaccard Distance was used to measure the predicate-level similarity between the human-generated logic clauses. For example, the Levenshtein Distance, the Jaccard Distance, and the Jaro Winkler Similarity between the two sample logic clauses in Table 1 were 14, 0.67, and 0.97, respectively. Overall, annotators reached an average Levenshtein Distance of 6.88, an average Jaccard Distance of 0.63, and an average Jaro Winkler Similarity of 0.99.

Table 1. Sample Logic Clauses Generated by Annotators

Building Code Sentence	Logic Clause 1	Logic Clause 2
The maximum width of a swinging door leaf shall be 48 inches (1219 mm) nominal.	compliance_width_of_swinging_door_leaf257(Swinging_door_leaf):-width(Width),swinging_door_leaf(Swinging_door_leaf),has(Swinging_door_leaf,Width),**less_than_or_equal_to**(Width,quantity(48,inches)).	compliance_width_of_swinging_door_leaf257(Swinging_door_leaf):-width(Width),swinging_door_leaf(Swinging_door_leaf),has(Swinging_door_leaf,Width),**equal_to**(Width,quantity(48,inches)).

Because text similarity is task-specific, there was no universally applicable standard for it. Instead, NLP researchers developed their own metrics according to the needs of tasks [80,81]. The 14 Levenshtein Distance seems high, but it does not consider the length of the text string. If the length of text string is considered, the 0.97 Jaro Winkler Similarity proves that human-generated logic clauses are similar at the character level. The 0.67 Jaccard Distance is relatively low. It indicates a significant number of predicates are different in human-generated logic clauses. However, the difference could be overstated because the Jaccard Distance requires two predicates to be completely identical in order to be considered the same. If two predicates are off by even one character, they are still considered different and accounted for in the Jaccard Distance. Combining the use of all three measures illustrates that human-generated logic clauses are similar in general. If

two predicates are different, they usually only differ by a few characters. Measurement results using the three metrics show that the three annotators reached a reasonable alignment and the quality of the gold standard was good [82,83].

5. Result

To evaluate the performance of the ruleset expansion method, the machine-generated logic clauses were compared against the human-generated gold standard. The closer the machine-generated logic clauses are to the gold standard, the better the pattern matching-based rules are, and therefore, the better performance the ruleset expansion method is deemed to have. Predicates in logic clauses can be further broken down into predicate elements (i.e., predicate names or predicate arguments). For example, the predicate "has(Stairway, Clear_width)" is formed by three elements, "has," "Stairway," and "Clear_width." Therefore, the authors calculated both predicate-level performance and predicate element-level performance of the ruleset expansion method. In the predicate-level performance, the minimum unit of measurement is a predicate. In the predicate element-level performance, the minimum unit of measurement is a word or phrase. For example, if the gold standard is "considered_by(Code_change_proposals, International_fire_code_development_committee)," and the machine-generated logic clause is "considered_by(Designation_f, International_fire_code_development_committee)." The predicate-level performance treats this predicate as one incorrect predicate. The predicate element-level performance treats this predicate as three elements, two of which are correct and one is incorrect. Because of the phenomenon that predicates could be partially correct. Predicate element-level accuracy could provide a more accurate evaluation regarding the performance of the ruleset expansion method and pattern matching-based rules.

The performance of the expanded pattern matching-based regulatory information transformation ruleset is summarized in Table 2. The performance was measured at the predicate level and the predicate element level. The experiment focuses on logic clauses about quantitative requirements because the original ruleset focused on quantitative requirements. In this research, sentences of building code provisions and generated logic clauses have a one-to-many mapping relationship. Patterns, on the other hand, can match the whole sentence or part of a sentence. Regulatory information that spans over multiple sentences is represented by multiple logic clauses. The original ruleset filters quantitative and non-quantitative requirements automatically. Therefore, there is no completely missed logic clause. The logic clauses generated that were not in the gold standard were counted as false positives. The logic clauses generated that functioned in the same way as those in the gold standard were counted as true positives. The logic clause-level performance is reported in Table 3. The predicate element-level performance was higher than the predicate-level performance, which indicates some predicates were partially correct. Through error analysis, the authors recognized four main sources of errors:

1. The partially correct predicates. After reviewing machine-generated logic clauses and the gold standard, the authors found that a significant portion of predicates in machine-generated logic

clauses were partially correct. For example, when the correct predicate in the gold standard is "surrounded_by(Buildings,Public_ways)," the expanded ruleset generated a partially correct predicate "surrounded_by(Chapter_9,Public_ways)." Future development about pattern matching-based rules could focus more on capturing the correct elements in predicates.

2. The flexibility of B-Prolog logic clauses and the ambiguity of natural language. The flexibility of B-Prolog logic clauses and the inherent ambiguity of natural language left a room of interpretation for the annotators. In other words, it was possible to represent the same building code requirement in different predicates. For example, one annotator translated the "minimum fire resistant rating of 1 hour" to "greater_than(Minimum_fire_resistance_rating, quantity(1, hour))." Another annotator translated the same phrase to "equal to (Minimum_fire_resistance_rating, quantity(1, hour))." One annotator considered that fire resistant rating of a subject should be greater than the minimum fire-resistant rating, which is 1 hour. Another annotator considered that this phrase means the minimum fire-resistant rating of a subject should be 1 hour. Therefore, the precision of the rule generation was affected. A viable solution to increase inter-annotator agreement may include more detailed and stricter annotation guidelines.
3. The patterns and terminologies unseen in Chapter 10. Although most regulatory information patterns in Chapter 5 were captured in Chapter 10, a small amount of regulatory information patterns were missed. The authors attributed missed building code requirements to unseen patterns or unique terminologies in Chapter 5. Chapter 5 and Chapter 10 of the IBC 2015 focus on different topics (i.e., general building height and area, and means of egress, respectively), so some terminologies in Chapter 5 did not occur in Chapter 10. For example, the erroneous predicate "unobstructed_to(Be,Room)" was generated instead of "unobstructed_to(Room)" due to a pattern that did not occur in Chapter 10. Such error will need to be addressed by accumulatively expanding the training dataset.
4. The backward compatibility requirement. The generality and validity requirements of the ruleset expansion method ensures the quality of the generated logic clauses and shows a promising future that pattern matching rule-based regulatory information extraction can potentially capture all building code requirements with a sufficiently comprehensive set of rules. However, the compatibility requirement also forbade the removal of any existing rule, which led to some false positives. In the future, the flexibility of modifying existing rules may need to be tested.

Table 2. Performance of Applying Ruleset Expansion Method

	Training				Testing			
	Predicate level		Predicate element level		Predicate level		Predicate element level	
	Before[1]	After[2]	Before[1]	After[2]	Before[1]	After[2]	Before[1]	After[2]
Precision	84.35%	96.31%	87.90%	98.33%	86.17%	95.17%	90.03%	97.48%
Recall	79.00%	98.38%	81.78%	99.39%	76.84%	96.60%	81.77%	98.65%
F1-score	81.59%	97.34%	84.73%	98.86%	81.24%	95.88%	85.70%	98.06%

[1] Performance of the original ruleset, which is the ruleset before the application of the ruleset expansion method.
[2] Performance of the expanded ruleset, which is the ruleset after the application of the ruleset expansion method.

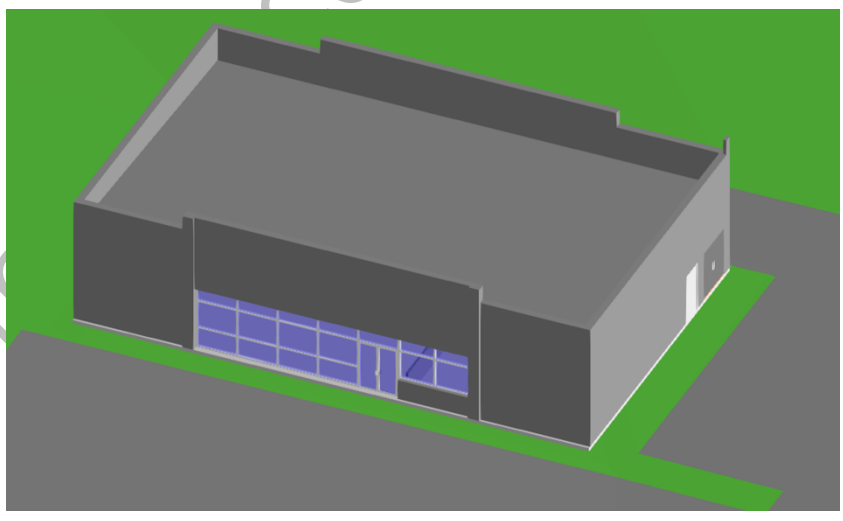
570 Table 3. Logic Clause-Level Performance

	Training		Testing	
	Before[1]	After[2]	Before[1]	After[2]
Precision	89.86%	100.00%	93.81%	100.00%
Recall	98.27%	99.68%	96.81%	97.98%
F1-score	93.87%	99.84%	95.29%	98.98%

[1] Performance of the original ruleset, which is the ruleset before the application of the ruleset expansion method.
[2] Performance of the expanded ruleset, which is the ruleset after the application of the ruleset expansion method.

571 # 6. Real-World Case Study

572 The proposed ruleset expansion method was also evaluated in a real-world test case (Figure 4). The
573 test case was based on a convention store in Texas to be checked against Chapter 5 and Chapter 10
574 of the IBC 2015. The BIM model and compliance checking report for the convention store were
575 provided by an industry partner of this research. According to the compliance checking report, five
576 applicable logic clauses in those sections were selected in the test case. Building design information
577 was extracted from the BIM model of the test case and further elaborated to fit the needs of the case
578 study. To make the test case comprehensive, three non-compliance instances with the applicable
579 logic clauses were added into the design information. In the test run, all three non-compliance
580 instances were successfully detected. Example checked logic clauses and corresponding non-
581 compliance instances are shown in Table 4.



582

583 **Figure 4. A Real-World Test Case for ACC.**

584

585 Table 4. Checked Logic Clauses

Section in IBC 2015	Building code requirement	Logic clause	Non-compliance instance
1010	"The required capacity of each door opening shall be sufficient for the occupant load thereof and shall provide a minimum clear width of 32 inches (813 mm)." [1]	compliance_door_opening(Door_opening):- required_capacity(Required_capacity),door_opening(Door_opening),has(Door_opening,Required_capacity),sufficient(Door_opening),for(Door_opening,Occupant_load),occupant_load(Occupant_load),provide(Required_capacity,Minimum_clear_width),minimum_clear_width(Minimum_clear_width),equal_to(Minimum_clear_width,quantity(32,inches)).).	A door opening provided a clear width of 27 inches.
1010	"The height of door openings shall be not less than 80 inches (2032mm)." [1]	compliance_height_of_door_openings(Door_openings):- height(Height),door_openings(Door_openings),has(Door_openings,Height),not less_than(Height,quantity(80,inches)).	The height of a door opening is 72 inches.
506	"To qualify for an area factor increase based on frontage, the public way or open space adjacent to the building perimeter shall have a minimum distance (W) of 20 feet (6096 mm) measured at right angles from the building face to any of the following:" [1]	compliance_Frontage(Frontage):- qualify_for(Qualify,Area_factor_increase),qualify(Qualify),area_factor_increase(Area_factor_increase),based_on(Area_factor_increase,Frontage),(frontage(Frontage);public_way(Frontage);open_space(Frontage)),minimum_distance(Minimum_distance),has(Building_perimeter,Minimum_distance),measured_at(Frontage,Right_angles),right_angles(Right_angles),from(Right_angles,Building_face),building_face(Building_face),equal_to(Minimum_distance,quantity(20,feet)),associated(Frontage,Minimum_distance).	The distance from the building face is 18 feet.

7. Robustness

In the experiment illustrated above, the proposed method was tested for expanding the range of checkable building code requirements to new chapters of building codes, which are in different domains/topics of the building code from which the original ruleset was initially developed. To further evaluate the robustness of the expanded ruleset, the authors also tested it on processing construction contracts, a fundamentally different type of construction documents compared to building codes. Nine free and openly available construction contracts or construction contract templates were collected. In total, 185 sentences were extracted from these contracts. The expanded ruleset was then executed to convert the extracted sentences in construction contracts to logic clauses. The performance of the expanded ruleset is illustrated in Table 5. Examples of contract contents and corresponding logic clauses generated are shown in Table 6. The results show the

robustness of the expanded ruleset is promising (although not perfect) for processing construction documents beyond the original intent of building codes.

Table 5. Performance on Processing Construction Contract

	Predicate level	Predicate element level
Precision	90.52%	97.20%
Recall	92.92%	98.42%
F1-score	91.70%	97.81%

Table 6. Examples of Contract Sentences and Corresponding Logic Clauses Generated

Contract sentence	Logic clause
Two copies of the Contract Documents shall be signed by the Owner and the Contractor.[84]	compliance_Owner3(Owner):-number_prep(Number),copies(Copies),has(Copies,Number),contract_documents(Contract_Documents),has(Contract_Documents,Copies),(owner(Owner);contractor(Owner)),signed_by(Contract_Documents,Owner),equal_to(Number,quantity(2,one)).
Contractor shall maintain in a safe place at the Property one record copy of all drawings, specifications, addenda, written amendments, and the like in good order and annotated to show all changes made during construction, which will be delivered to Owner upon completion of the Work. [85]	compliance_Number1(Number):-maintain_in(Contractor,Safe_place),contractor(Contractor),safe_place(Safe_place),at(Safe_place,Property),property(Property),number_prep(Number),record_copy(Record_copy),has(Record_copy,Number),drawings(Drawings),has(Drawings,Record_copy),like_in(Like,Good_order),like(Like),good_order(Good_order),to_show(Good_order,Changes),changes(Changes),made_during(Changes,Construction),construction(Construction),delivered_to(Good_order,Owner),owner(Owner),upon(Owner,Completion),completion(Completion),work(Work),has(Work,Completion),equal_to(Number,quantity(1,one)),associated(Safe_place,Good_order).
If the final amount of the ALLOWANCE work is less than the ALLOWANCE line item amount listed in the Agreement, a credit will be issued to Owner after all billings related to this particular line item ALLOWANCE work have been received by Contractor. [86]	compliance_Final_amount7(Final_amount):-final_amount(Final_amount),if(Final_amount),allowance_work(ALLOWANCE_work),has(ALLOWANCE_work,Final_amount),allowance_line_item_amount(ALLOWANCE_line_item_amount),listed_in(ALLOWANCE_line_item_amount,Agreement),agreement(Agreement),credit(Credit),owner(Owner),issued_to(Credit,Owner),after(Owner,Billings),billings(Billings),related_to(Billings,This_particular_line_item_ALLOWANCE_work),this_particular_line_item_allowance_work(This_particular_line_item_ALLOWANCE_work),received_by(This_particular_line_item_ALLOWANCE_work,Contractor),contractor(Contractor),less_than(Final_amount,quantity(1,ALLOWANCE_line_item_amount)).

8. Contributions to the Body of Knowledge

This research contributes to the body of knowledge in four main ways. First, this research proves the feasibility of expanding the range of checkable building code requirements by expanding an existing regulatory information transformation ruleset. The authors expanded the range of checkable building code requirements of an automated code compliance checking system to cover Chapter 5 and Chapter 10 of the IBC 2015. This expansion was achieved by 64 new rules. It shows that different chapters of the IBC share similar patterns, and the number of new pattern matching-based

rules needed to expand the range of checkable code requirements is small. Second, this research proposed a new ruleset expansion method. This method ensures the quality of added pattern-matching-based rules and, therefore, the quality of logic clauses generated by the pattern matching-based rules. In a previous study, three hundred and six rules were developed to cover two chapters of building code. In comparison, only sixty-four new rules were developed to cover two new chapters of building code. This research shows that the marginal cost of expanding the range of checkable building code requirements is low. It provides a workable and low-cost method to expand the range of checkable code requirements of ACC systems. The cost of expanding the range of checkable building codes by expanding an existing regulatory information transformation ruleset could further decrease in the future as the number of existing rules increases, because building codes share similar patterns and the number of unseen patterns in new building codes could decrease as existing pattern matching-based rules cover more patterns in building codes. Future researchers and developers can adopt this method to expand the range of checkable code requirements of the ACC system and bring the ACC system to full deployment in the AEC industry. While the research focuses on processing building codes, the testing results of transforming construction contracts show that the proposed ruleset expansion method is potentially robust in processing different types of construction documents. Third, this research also generated a dataset of building codes in logic clauses. This dataset can facilitate other regulatory information transformation research, such as machine learning-based logic clause generation. Last but not least, this research facilities the adoption of ACC in the AEC industry. With an expanded range of checkable code requirements, the utility of ACC is enhanced. ACC can reduce the time, cost, and human-errors in code compliance checking and encourages the AEC industry to shift towards a digital paradigm.

9. Conclusion

The paper presents a ruleset expansion method that can extend the range of checkable code requirements of ACC systems to different chapters of the International Building Code (IBC), which can potentially be applied to other codes beyond the IBC and other construction documents such as contracts. It takes an iterative approach to ensure the generality and validity of the added pattern matching-based rules and the generated logic clauses. Experimental results on Chapters 5 and 10 of IBC 2015 showed the expanded ruleset generated logic clauses with 95.17% predicate-level precision, 96.60% predicate-level recall, and 95.88% predicate-level F1-score. This performance proved the effectiveness of the ruleset expansion method and the expanded ruleset. Through error analysis, the authors attributed the remaining errors to the flexibility of B-Prolog language, the ambiguity in natural language, missed building code requirement patterns, and the compatibility requirement. The authors also suggested solutions to further increase the performance of the ruleset expansion method, such as expanding the training dataset and providing stricter annotation guidelines. This research also presents a dataset of logic clauses. This dataset has the potential to facilitate research on different regulatory information transformation approaches, such as machine learning-based logic clause generation. The research findings in this study can be used to build fully

automated building code compliance checking systems with wider code requirements coverage than the state of the art. The demonstrated decreasing marginal cost of transformation rule development and high predicate-level performance renders the rule-based processing of building code requirements promising to bring fully automated building code compliance checking to real-world applications. Future research is needed to discover the boundary of the theoretical "superset" of common patterns used in building code transformation rules, for guiding the practical implementation of the demonstrated rule-based processing of building code requirements in real ACC systems. Furthermore, the successful demonstration of such processing in construction contracts in this study helps open the door to rule-based processing of a variety of textual documents in the AEC industry, to support future automation and AI applications in the AEC industry.

10. Acknowledgement

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.

11. Reference

[1] International Code Council, International Building Code, International Code Council,, 2015.1609837398, https://codes.iccsafe.org/content/IBC2015 , Last Access: Feb 12th, 2022.

[2] International Code Council, International Fire Code, International Code Council, 2015.1609837398, https://codes.iccsafe.org/content/IFC2015, Last Access: Feb 12th, 2022..

[3] Indiana Department of Homeland Security, Codes, Standards and Other Rules, in: Secondary Indiana Department of Homeland Security (Ed.), Secondary Codes, Standards and Other Rules, https://www.in.gov/dhs/2490.htm#top, Last Access: Jan 21st,2020.

[4] S. Moon, G. Lee, S. Chi, H. Oh, Automatic Review of Construction Specifications Using Natural Language Processing, in: Proceedings of Computing in Civil Engineering 2019: Data, Sensing, and Analytics, American Society of Civil Engineers, Atlanta, Georgia, 2019, pp. 401-407. https://doi.org/10.1061/9780784482438.051.

[5] C. Preidel, A. Borrmann, Refinement of the visual code checking language for an automated checking of building information models regarding applicable regulations, in: Proceedings of Computing in Civil Engineering 2017, American Society of Civil Engineers, Seattle, Washington, 2017, pp. 157-165. https://doi.org/10.1061/9780784480823.020.

[6] A. Alghamdi, M. Sulaiman, A. Alghamdi, M. Alhosan, M. Mastali, J. Zhang, Building accessibility code compliance verification using game simulations in virtual reality, Computing in Civil Engineering 2017, Seattle, Washington, 2017, pp. 262-270. https://doi.org/10.1061/9780784480830.033.

[7] City of Chicago, Average Time for Permit Issuance, in: Secondary City of Chicago (Ed.), Secondary Average Time for Permit Issuance, https://www.chicago.gov/city/en/depts/bldgs.html, Last Access: Jan, 21st,2020.

[8] City of Chicago, Permit Fee Calculator, in: Secondary City of Chicago (Ed.), Secondary Permit Fee Calculator, https://ipiweb.cityofchicago.org/DynamicPortal/Forms/FeeCalculator.aspx, Last access: Last Access: Jan, 21st,2020.

[9] X. Tan, A. Hammad, P. Fazio, Automated code compliance checking for building envelope design, Journal of Computing in Civil Engineering 24 (2) (2010) pp. 203-211, https://doi.org/10.1061/(ASCE)0887-3801(2010)24:2(203).

[10] J. Martins, V. Abrantes, Automated code-checking as a driver of BIM adoption, International Journal for Housing Science, 34 (2010) pp. 287-295, http://www.housingscience.org/html/publications/pdf/34-4-6.pdf, Last Access: Feb 12th, 2022.

[11] D. Greenwood, S. Lockley, S. Malsane, J. Matthews, Automated compliance checking using building information models, in: Proceedings of the The Construction, Building and Real Estate Research Conference of the Royal Institution of Chartered Surveyors, Royal Institution of Chartered Surveyors (RICS), Paris, France, 2010, pp. 266-274. https://www.irbnet.de/daten/iconda/CIB20057.pdf.

[12] J. Choi, I. Kim, An approach to share architectural drawing information and document information for automated code checking system, Tsinghua Science and Technology 13 (S1) (2008) pp. 171-178, https://doi.org/10.1016/S1007-0214(08)70145-7.

[13] J. Zhang, N.M. El-Gohary, Integrating semantic NLP and logic reasoning into a unified system for fully-automated code checking, Automation in Construction 73 (2017) pp. 45-57, https://doi.org/10.1016/j.autcon.2016.08.027.

[14] S.J. Fenves, Tabular decision logic for structural design, Journal of the Structural Division 92 (6) (1966) pp. 473-490, https://doi.org/10.1061/JSDEAG.0001567.

[15] J.H. Garrett, S.J. Fenves, A knowledge-based standards processor for structural component design, Engineering with Computers 2 (4) (1987) pp. 219-238, https://doi.org/10.1007/BF01276414.

[16] L. Lopez, S. Elam, K. Reed, Software concept for checking engineering designs for conformance with codes and standards, Engineering with Computers 5 (2) (1989) pp. 63-78, https://doi.org/10.1007/BF01199070.

[17] E.A. Delis, A. Delis, Automatic fire-code checking using expert-system technology, Journal of Computing in Civil Engineering 9 (2) (1995) pp. 141-156, https://doi.org/10.1061/(ASCE)0887-3801(1995)9:2(141).

[18] J.S. Gero, Design prototypes: a knowledge representation schema for design, AI Magazine 11 (4) (1990) pp. 26-36, https://doi.org/10.1609/aimag.v11i4.854.

[19] L. Ding, R. Drogemuller, M. Rosenman, D. Marchant, J. Gero, Automating code checking for building designs-DesignCheck, in: Proceedings of the Construction Research Congress (CRC) for Construction Innovation. American Society of Civil Engineerings, Grand Bahama Island, Bahamas, 2006 pp. 1-16, https://doi.org/10.1108/ecam.2006.28613faa.002, Last Access: Feb 12th, 2022.

[20] J. Dimyadi, R. Amor, Automated building code compliance checking–where is it at?, in: Proceedings of the 19th International World Building Congress, (6), International Council for Research and Innovation in Building and Construction, Brisbane, Australia, 2013. https://doi.org/10.13140/2.1.4920.4161.

[21] A.R. Sabouni, O.M. Al-Mourad, Quantitative knowledge based approach for preliminary design of tall buildings, Artificial Intelligence in Engineering 11 (2) (1997) pp. 143-154, https://doi.org/10.1007/s00216-016-0157-x.

[22] C. Eastman, J.M. Lee, Y.S. Jeong, J.K. Lee, Automatic rule-based checking of building designs, Automation in Construction 18 (8) (2009) pp. 1011-1033, https://doi.org/10.1016/j.autcon.2009.07.002 .

[23] Y. Shao, C. Hardmeier, J. Tiedemann, J. Nivre, Character-based joint segmentation and POS tagging for Chinese using bidirectional RNN-CRF, arXiv preprint arXiv:1704.01314 (2017), https://aclanthology.org/I17-1018, Last Access: Feb 12th, 2022.

[24] B.T. Zhong, L.Y. Ding, H.B. Luo, Y. Zhou, Y.Z. Hu, H.M. Hu,Ontology-based semantic modeling of regulation constraint for automated construction quality compliance checking, Automation in Construction 28 (2012) pp. 58-70, https://doi.org/10.1016/j.autcon.2012.06.006.

[25] J. Zhang, N.M. El-Gohary, Semantic NLP-based information extraction from construction regulatory documents for automated compliance checking, Journal of Computing in Civil Engineering 30 (2) (2016) pp. 04015014, https://doi.org/10.1061/(ASCE)CP.1943-5487.0000346.

[26] J. Dimyadi, G. Clifton, M. Spearpoint, R. Amor, Computerizing Regulatory Knowledge for Building Engineering Design, Journal of Computing in Civil Engineering (2016) pp. C4016001, https://doi.org/10.1061/(ASCE)CP.1943-5487.0000572.

[27] S.M. İlal, H.M. Günaydın, Computer representation of building codes for automated compliance checking, Automation in Construction 82 (2017) pp. 43-58, https://doi.org/10.1016/j.autcon.2017.06.018.

[28] T.H. Nguyen, Integrating building code compliance checking into a 3D CAD system, in: Proceedings of the Computing in Civil Engineering (2005), American Society of Civil Engineers, Cancun, Mexico, 2005, pp. 1-12. https://doi.org/10.13140/2.1.4920.4161.

[29] N.O. Nawari, Automating codes conformance, Journal of Architectural Engineering 18 (4) (2012) pp. 315-323, https://doi.org/10.1061/41182(416)70.

[30] J. Zhang, N.M. El-Gohary, Semantic-based logic representation and reasoning for automated regulatory compliance checking, Journal of Computing in Civil Engineering 31 (1) (2017) pp. 04016037, https://doi.org/10.1061/(ASCE)CP.1943-5487.0000583.

[31] J. Zhang, N.M. El-Gohary, Automated information transformation for automated regulatory compliance checking in construction, Journal of Computing in Civil Engineering 29 (4) (2015) pp. B4015001, https://doi.org/:10.1061/(ASCE)CP.1943-5487.0000427.

[32] J. Zhang, N.M. El-Gohary, Automated extraction of information from building information models into a semantic logic-based representation, Congress on Computing in Civil Engineering, Proceedings 2015 (2015) pp. 173-180,

https://doi.org/10.1061/9780784479247.022.

[33] Zhang, J., and El-Gohary, N. Integrating semantic NLP and logic reasoning into a unified system for fully-automated code checking. Automation in Construction, 73, pp. 45-57, https://doi.org/10.1016/j.autcon.2016.08.027.

[34] Zhang, J. and El-Gohary, N. Semantic-based logic representation and reasoning for automated regulatory compliance checking. Journal of Computing in Civil Engineering, 31 (1), pp.4016037, https://doi.org/10.1061/(ASCE)CP.1943-5487.0000583.

[35] S.A. Morshed, X. Lv, R.B. Tanvir, Network-based information extraction from IFC files to support intelligent BIM companion (iBcom) technology, in: Proceedings of Construction Research Congress 2020, American Society of Civil Engineers, Tempe, Arizona, 2020, pp. 427-435. https://doi.org/10.1061/9780784482865.046.

[36] P. Zhou, N. El-Gohary, Automated matching of design information in BIM to regulatory information in energy codes, in: Proceedings of the Construction Research Congress 2018, American Society of Civil Engineers, New Orleans, Louisiana, 2018, pp. 75-85. https://doi.org/10.1061/9780784481264.008.

[37] G.G. Chowdhury, Natural language processing, Annual review of information science and technology 37 (1) (2003) pp. 51-89, https://doi.org/10.1002/aris.1440370103.

[38] C.D. Manning, P. Raghavan, H. Schütze, Introduction to information retrieval, Cambridge University Press, 2008.0521865719, https://wiltrud.hwro.de/teaching/ir12w/pdf/19web.pdf, Last Access: Feb 12th, 2022..

[39] J. Cowie, W. Lehnert, Information extraction, Communications of the Association for Computing Machinery (ACM), 39 (1) (1996) pp. 80-91, https://doi.org/10.1145/234173.234209.

[40] X. Zhang, J. Zhao, Y. LeCun, Character-level convolutional networks for text classification, in: Proceedings of the 28th International Conference on Neural Information Processing Systems, Association for Computing Machinery (ACM), Montreal, Canada, 2015, pp. 649-657. https://dl.acm.org/doi/10.5555/2969239.2969312.

[41] K. McKeown, Text generation, Cambridge University Press, 1992.0521438020, https://www.cambridge.org/core/books/text-generation/EC9075AB5F64A7AFDC5A9513237505CA, Last Access: Feb 12th, 2022.

[42] A. Nenkova, K. McKeown, A Survey of Text Summarization Techniques, Mining Text Data (2012) pp. 43-76, https://doi.org/10.1007/978-1-4614-3223-4_3.

[43] R. Barzilay, M. Elhadad, Using lexical chains for text summarization, Advances in Automatic Text Summarization (1999) pp. 111-121, https://aclanthology.org/W97-0703, ,Last Access: Feb 12th, 2022.

[44] P. Koehn, Statistical machine translation, Cambridge University Press, 2009.1139483307, https://www.cambridge.org/core/books/statistical-machine-translation/94EADF9F680558E13BE759997553CDE5, Last Access: Feb 12th, 2022.

[45] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, The Kaldi speech recognition toolkit, in: Proceedings of the Institute of Electrical and Electronics Engineers (IEEE)2011 Workshop on Automatic Speech Recognition and Understanding, Institute of Electrical and Electronics Engineers (IEEE)Signal

Processing Society, New York, New York, 2011. https://www.danielpovey.com/files/2011_asru_kaldi.pdf, Last Access: Feb 12th, 2022.

[46] K. Gali, H. Surana, A. Vaidya, P.M. Shishtla, D.M. Sharma, Aggregating machine learning and rule based heuristics for named entity recognition, in: Proceedings of the International Joint Conference on Natural Language Processing-08 (IJCNLP) Workshop on Named Entity Recognition for South and South East Asian Languages, Association for Computational Linguistics, Hyderabad, India, 2008, https://www.aclweb.org/anthology/I08-5005, Last Access: Feb 12th, 2022.

[47] K. Crowston, X. Liu, E.E. Allen, Machine learning and rule-based automated coding of qualitative data, in: Proceedings of the American Society for Information Science and Technology, (47), Association for Information Science and Technology, Pittsburgh, Pennsylvania, 2010, pp. 1-2. https://doi.org/10.1002/meet.14504701328.

[48] F. Chollet, Deep Learning with Python, 2017.1617294438, https://www.manning.com/books/deep-learning-with-python, ,Last Access: Feb 12th, 2022.

[49] A.J.P. Tixier, M.R. Hallowell, B. Rajagopalan, D. Bowman, Automated content analysis for construction safety: A natural language processing system to extract precursors and outcomes from unstructured injury reports, Automation in Construction 62 (2016) pp. 45-56, https://doi.org/10.1016/j.autcon.2015.11.001.

[50] J.R. Lin, Z. Hu, J. Zhang, BIM oriented intelligent data mining and representation, in: Proceedings of 30th International Council for Research and Innovation in Building and Construction W78 International Conference on Applications of IT in the AEC Industry, International Council for Research and Innovation in Building and Construction, Beijing, China, 2013, pp. 280-289. https://itc.scix.net/pdfs/w78-2013-paper-112.pdf, Last Access: Feb 12nd, 2022 .

[51] R. Zhang, N. El-Gohary, A machine-learning approach for emantic matching of building codes and Building Information Models (BIMs) for supporting automated code checking, in: the Proceedings of the International Congress and Exhibition "Sustainable Civil Infrastructures", Springer, Egypt, Egypt, 2019, pp. 64-73. https://doi.org/10.1007/978-3-030-34216-6_5.

[52] J. Zhang, N. El-Gohary, Extending building information models semiautomatically using semantic natural language processing techniques, Journal of Computing in Civil Engineering (2016) pp. C4016004, https://doi.org/10.1061/(ASCE)CP.1943-5487.0000536.

[53] E. Brill, A simple rule-based part of speech tagger, in: the Proceedings of the Third Conference on Applied Natural Language Processing, Association for Computational Linguistics, Trento, Italy 1992, pp. 152-155. https://doi.org/10.3115/974499.974526.

[54] Butte College, The eight parts of speech, in: Secondary Butte College (Ed.), Secondary, The Eight Parts of Speech, http://www.butte.edu/departments/cas/tipsheets/grammar/parts_of_speech.html, Last Access: Sep 11st,2019.

[55] M. Marcus, B. Santorini, M.A. Marcinkiewicz, Building a large annotated corpus of English: The Penn Treebank, (1993), https://doi.org/10.5555/972470.972475, Last Access: Feb 12nd,

2022.

[56] W.N. Francis, H. Kucera, Brown Corpus Manual, Brown University, 1979, http://korpus.uib.no/icame/manuals/BROWN/INDEX.HTM, Last Access: Feb 12nd, 2022.

[57] H. Schmid, Part-of-speech tagging with neural networks, in: the Proceedings of the 15th conference on Computational linguistics, (1), Association for Computational Linguistics, Kyoto, Japan, 1994, pp. 172-176. https://doi.org/10.3115/991886.991915.

[58] S. Bird, E. Klein, E. Loper, Natural language processing with Python: analyzing text with the natural language toolkit, O'Reilly Media, Inc., 2009.0596555717, https://www.nltk.org/book/, Last Access: Feb 12nd, 2022.

[59] J. Giménez, L. Marquez, Fast and accurate part-of-speech tagging: The SVM approach revisited, in: the Proceedings of the Recent Advances in Natural Language Processing III, Association for Computational Linguistics, Borovets, Bulgaria, 2003, pp. 153-162. https://dblp.org/rec/conf/ranlp/GimenezM03.

[60] T. Brants, TnT: a statistical part-of-speech tagger, in: the Proceedings of the sixth conference on Applied natural language processing, Association for Computational Linguistics, Seattle, Washington, 2000, pp. 224-231. https://doi.org/10.3115/974147.974178.

[61] B. Plank, A. Søgaard, Y. Goldberg, Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, (2), Association for Computational Linguistics, Berlin, Germany, 2016, pp. 412–418. https://doi.org/10.18653/v1/P16-2067.

[62] T.R. Gruber, A translation approach to portable ontology specifications, Knowledge Acquisition 5 (2) (1993) pp. 199-221, https://doi.org/10.1006/knac.1993.1008.

[63] D. Brickley, R.V. Guha, A. Layman, Resource description framework (RDF) schema specification, Encyclopedia of Database Systems,(1999), https://doi.org/10.1007/978-0-387-39940-9_1319.

[64] J. Hendler, D.L. McGuinness, The DARPA agent markup language, in: Proceedings of Institute of Electrical and Electronics Engineers (IEEE) Intelligent Systems, (15), Institute of Electrical and Electronics Engineers (IEEE), New York, New York, 2000, pp. 67-73. http://www-ksl.stanford.edu/pub/KSL_Reports/KSL-00-10.html, Last Access: Feb 12nd, 2022.

[65] D.L. Mcguinness, R. Fikes, J. Hendler, L.A. Stein, DAML+OIL: an ontology language for the Semantic Web, in: Proceedings of Institute of Electrical and Electronics Engineers (IEEE)Intelligent Systems 17 (5), Institute of Electrical and Electronics Engineers (IEEE), New York, New York, 2002 pp. 72-80, https://doi.org/10.1109/MIS.2002.1039835.

[66] L. Amos, D. Anderson, S. Brody, A. Ripple, B.L. Humphreys, UMLS users and uses: a current overview, Journal of the American Medical Informatics Association 27 (10) (2020) pp. https://doi.org/1606-1611, 10.1093/jamia/ocaa084.

[67] M. Hepp, Goodrelations: An ontology for describing products and services offers on the web, in: Proceeding of the International Conference on Knowledge Engineering and Knowledge Management, Springer, Berlin, Heidelberg, 2008, pp. 329-346. https://doi.org/10.1007/978-3-540-87696-0_29.

[68] N.F. Noy, D.L. McGuinness, Ontology development 101: A guide to creating your first ontology,

in: Secondary N.F. Noy, D.L. McGuinness (Eds.), Secondary Ontology development 101: A guide to creating your first ontology, http://www.ksl.stanford.edu/people/dlm/papers/ontology-tutorial-noy-mcguinness-abstract.html, Last Access: May 22nd,2021

[69] W.H. Gomaa, A.A. Fahmy, A survey of text similarity approaches, International Journal of Computer Applications 68 (13) (2013) pp. 13-18, https://doi.org/10.5120/11638-7118.

[70] Z. Su, B. Ahn, K. Eom, M. Kang, J. Kim, M. Kim, Plagiarism Ddetection using the Levenshtein Distance and Smith-Waterman Algorithm, in: Proceeding of the 2008 3rd International Conference on Innovative Computing Information and Control, Institute of Electrical and Electronics Engineers (IEEE), Dalian, China, 2008, pp. 569-569. https://doi.org/10.1109/ICICIC.2008.422.

[71] V.I. Levenshtein, Binary codes capable of correcting deletions, insertions, and reversals, Soviet Physics Doklady, (10), 1966, pp. 707-710. https://ui.adsabs.harvard.edu/abs/1966SPhD...10..707L/abstract, Last Access: Feb 12nd, 2022.

[72] S. Kosub, A note on the triangle inequality for the Jaccard distance, Pattern Recognition Letters 120 (2019) pp. 36-38, https://doi.org/10.1016/j.patrec.2018.12.007.

[73] W.E. Winkler, String comparator metrics and enhanced decision rules in the Fellegi-Sunter sodel of record linkage,, https://eric.ed.gov/?id=ED325505, Last Access: May 22nd, 2021.

[74] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Proceedings of the 26th International Conference on Neural Information Processing Systems, (2), Curran Associates Inc., Lake Tahoe, Nevada, 2013, pp. 3111–3119. https://doi.org/10.5555/2999792.2999959.

[75] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, T. Mikolov, Fasttext. zip: Compressing text classification models, arXiv preprint arXiv:1612.03651 (2016), https://arxiv.org/abs/1612.03651.

[76] J. Pennington, R. Socher, C.D. Manning, Glove: Global vectors for word representation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Doha, Qatar, 2014, pp. 1532-1543. https://doi.org/10.3115/v1/D14-1162.

[77] A. Ben Abacha, P. Zweigenbaum, Automatic extraction of semantic relations between medical entities: a rule based approach, Journal of Biomedical Semantics 2 (5) (2011) pp. S4, https://doi.org/10.1186/2041-1480-2-S5-S4.

[78] C. Zhang, X. Zhang, W. Jiang, Q. Shen, S. Zhang, Rule-Based Extraction of Spatial Relations in Natural Language Text, in: Proceedings of the 2009 International Conference on Computational Intelligence and Software Engineering, Institute of Electrical and Electronics Engineers (IEEE), Washington, DC, 2009, pp. 1-4. https://doi.org/10.1109/CISE.2009.5363900.

[79] H. Cunningham, GATE, a general architecture for text engineering, Computers and the Humanities 36 (2) (2002) pp. 223-254, https://doi.org/10.1023/A:1014348124664.

[80] P. Penumatsa, M. Ventura, A.C. Graesser, M. Louwerse, X. Hu, Z. Cai, D.R. Franceschetti, The right threshold value: what is the right threshold of cosine measure when using latent semantic analysis for evaluating student answers?, International Journal on Artificial Intelligence Tools 15 (05) (2006) pp. 767-777, https://doi.org/10.1142/S021821300600293X.

[81] N. Rekabsaz, M. Lupu, A. Hanbury, Exploration of a threshold for similarity based on uncertainty in word embedding, in: Proceedings of the European Conference on Information Retrieval, Springer, Scotland, UK, 2017, pp. 396-409. https://doi.org/10.1007/978-3-319-56608-5_31.

[82] S. Cahyono, Comparison of document similarity measurements in scientific writing using Jaro-Winkler Distance method and Paragraph Vector method, in: Proceedings of the Institute of Physics (IOP)Conference Series: Materials Science and Engineering,(662), Institute of Physics (IOP)Publishing, London, UK, 2019, p. 052016. https://doi.org/10.1088/1757-899X/662/5/052016.

[83] I. Kloo, M.F. Dabkowski, S.H. Huddleston, Improving record linkage for counter-threat finance intelligence with dynamic Jaro-Winkler thresholds, in: Proceedings of the2019 Winter Simulation Conference (WSC), Institute of Electrical and Electronics Engineers (IEEE), National Harbor, Maryland, 2019, pp. 2467-2478. https://doi.org/10.1109/WSC40007.2019.9004945.

[84] Montrose County, , Construction Contract, https://www.montrosecounty.net/DocumentCenter/View/823/Sample-Construction-Contract, Last Access: December 21st, 2021.

[85] Legal Templates, Create a free construction contract agreement,https://legaltemplates.net/form/lt/construction-contract-agreement/?utm_source=google&utm_medium=cpc&utm_campaign=81-Construction+Contractor+Agreement&utm_term=free+construction+contracts&utm_campaign_id=806343792&utm_adgroup=Construction+Contractor+Agreement+-+Simplified&utm_adgroup_id=44606415274&utm_content=191412796236&gclid=Cj0KCQiAk4aOBhCTARIsAFWFP9FpR07y2Zo4j5TNjPUCraxOFmc68kb3LNpChyqYuDfo6jMpCYjUJIMaAqslEALw_wcB, Last Access: December 21st, 2021.

[86] K. Potts, Model construction contract for homeowners https://buildingadvisor.com/project-management/contracts/model-construction-contract_1/, Last Access: December 21st, 2021.

Appendix A: Patterns Used in Expanded Pattern Matching-based Rules

1. [complementary subject, candidate subject, candidate compliance checking attribute], inter clause boundary relation, [complementary subject, candidate subject, candidate compliance checking attribute], indicating "part_of" or "belongs_to" relation by the term "of", [complementary subject, candidate subject, candidate compliance checking attribute], Conjunctive Term, [complementary subject, candidate subject, candidate compliance checking attribute].

2. candidate subject, preposition, complementary subject, inter clause boundary relation, candidate subject, adjective, preposition, candidate subject.
3. [candidate subject, complementary subject, comparative relation], inter clause boundary relation, gerund or present participle verb, [candidate subject, complementary subject, comparative relation].
4. [complementary subject, candidate subject, candidate compliance checking attribute], past participle verb, comparative relation, value, unit, conjunctive term, comparative relation, value, unit.
5. complementary subject, modal verb, base form verb, value, unit, comparative relation, comparative relation, conjunctive term, value, unit, adjective, preposition, candidate subject.
6. [complementary subject, candidate subject, candidate compliance checking attribute], modal verb, negation, base form verb, comparative relation, value, unit, preposition, complementary subject.
7. [candidate subject, complementary subject, comparative relation], gerund or present participle verb, inter clause boundary relation, [candidate subject, complementary subject, comparative relation].
8. [candidate subject, complementary subject], inter clause boundary relation, [candidate subject, complementary subject], inter clause boundary relation
9. [candidate subject, complementary subject], slash "/", [candidate subject, complementary subject].
10. candidate compliance checking attribute, indicating "part_of" or "belongs_to" relation by the term "of", for each, candidate subject.
11. candidate subject, relation verb, inter clause boundary relation, candidate subject
12. candidate compliance checking attribute, modal verb, base form verb, negation, comparative relation, candidate compliance checking attribute.
13. value, complementary subject, preposition, for each, value, unit, indicating "part_of" or "belongs_to" relation by the term "of", candidate compliance checking attribute.
14. [candidate subject, complementary subject, candidate compliance checking attribute], preposition, for each..
15. [complementary subject, candidate subject, candidate compliance checking attribute], [non-3rd person singular present verb, modal verb, base form verb],possessive subject restriction, value,[complementary subject, candidate subject, candidate compliance checking attribute].
16. [complementary subject, candidate subject, candidate compliance checking attribute], [non-3rd person singular present verb, base form verb, 3rd person singular present verb], comparative relation, value, [complementary subject, candidate subject, candidate compliance checking attribute].
17. [candidate subject, complementary subject, comparative relation], inter clause

boundary relation, conjunctive term, [candidate subject, complementary subject, comparative relation].

18. complementary subject, preposition, complementary subject, modal verb, negation, base form verb, candidate compliance checking attribute.
19. [complementary subject, candidate subject, candidate compliance checking attribute], indicating "part_of" or "belongs_to" relation by the term "of", complementary subject, non-3rd person singular present verb,3rd person singular present verb, negation, comparative relation, value, unit.
20. [candidate subject, complementary subject, candidate compliance checking attribute], for "with" or "with in" relation, [candidate subject, complementary subject, candidate compliance checking attribute].
21. base form verb, preposition, [candidate subject, complementary subject, candidate compliance checking attribute.
22. candidate subject, modal verb, negation, base form verb, candidate subject.
23. [candidate subject, candidate compliance checking attribute], relation verb, [complementary subject, candidate compliance checking attribute], inter clause boundary relation, complementary subject.
24. [candidate subject, complementary subject, comparative relation], indicating "part_of" or "belongs_to" relation by the term "of", for each, [candidate subject, complementary subject, comparative relation].
25. complementary subject, character, cardinal number.
26. [candidate subject, candidate compliance checking attribute], indicating "part_of" or "belongs_to" relation by the term "of", value, unit, adjective.
27. candidate compliance checking attribute, gerund or present participle verb, inter clause boundary relation, [candidate subject, complementary subject, comparative relation, candidate compliance checking attribute].
28. [complementary subject, candidate subject, candidate compliance checking attribute], preposition, [complementary subject, candidate subject, candidate compliance checking attribute], [preposition, the word "to"], [complementary subject, candidate subject, candidate compliance checking attribute].
29. complementary subject, modal verb, base form verb, preposition, candidate subject, value.
30. candidate compliance checking attribute, modal verb, negation, base form verb, the word "to", candidate subject.
31. [complementary subject, candidate subject, candidate compliance checking attribute], relation verb, value, unit, conjunctive term, value, unit.
32. complementary subject, modal verb, base form verb, negation, comparative relation, value, unit, preposition, candidate compliance checking attribute.
33. candidate compliance checking attribute, conjunctive term, past participle verb, candidate compliance checking attribute.

34. [candidate subject, complementary subject, candidate compliance checking attribute, comparative relation], 3rd person singular present verb, past participle verb.
35. [complementary subject, candidate compliance checking attribute], modal verb, base form verb, relation verb, [complementary subject, candidate subject, candidate compliance checking attribute.
36. candidate subject, modal verb, negation, base form verb, candidate compliance checking attribute.
37. preposition, value, unit, candidate subject.
38. modal verb, negation, base form verb, [adjective, past participle verb, past tense verb]
39. value, unit, preposition, candidate subject.
40. preposition, value, unit, indicating "part_of" or "belongs_to" relation by the term "of", candidate subject.
41. relation verb, candidate compliance checking attribute, indicating "part_of" or "belongs_to" relation by the term "of", cardinal number, conjunctive term, comparative adjective.
42. preposition, past participle verb, candidate compliance checking attribute.
43. complementary subject, the word "to", value, complementary subject.
44. negation, comparative relation, value, slash "/", unit, candidate compliance checking attribute.
45. candidate subject, possessive subject restriction.
46. complementary subject, candidate compliance checking attribute.
47. preposition, value, unit, candidate subject, indicating "part_of" or "belongs_to" relation by the term "of", candidate compliance checking attribute.
48. complementary subject, modal verb, possessive subject restriction, complementary subject.
49. complementary subject, candidate subject, candidate compliance checking attribute], value, conjunctive term, comparative adjective, [complementary subject, candidate subject, candidate compliance checking attribute].
50. adjective, indicating "part_of" or "belongs_to" relation by the term "of", [candidate subject, complementary subject, candidate compliance checking attribute].
51. preposition, value, unit, preposition, candidate compliance checking attribute.
52. candidate compliance checking attribute, indicating "part_of" or "belongs_to" relation by the term "of", value, unit, the word "to", value, unit.
53. [candidate subject, complementary subject, candidate compliance checking attribute, comparative relation], indicating "part_of" or "belongs_to" relation by the term "of", gerund or present participle verb, [candidate subject, complementary subject, candidate compliance checking attribute, comparative relation].

54. comparative relation, value, [candidate subject, complementary subject].
55. [candidate subject, complementary subject, candidate compliance checking attribute, comparative relation], indicating "part_of" or "belongs_to" relation by the term "of", comparative adjective, [candidate subject, complementary subject, candidate compliance checking attribute, comparative relation].
56. complementary subject, candidate subject, candidate compliance checking attribute], negation, comparative relation, value, [complementary subject, candidate subject, candidate compliance checking attribute].
57. candidate subject, gerund or present participle verb, candidate compliance checking attribute, indicating "part_of" or "belongs_to" relation by the term "of", comparative relation, value.
58. negation, comparative relation, value, indicating "part_of" or "belongs_to" relation by the term "of", candidate compliance checking attribute, indicating "part_of" or "belongs_to" relation by the term "of", complementary subject.
59. candidate compliance checking attribute, modal verb, base form verb, past participle verb.
60. negation, comparative relation, value, indicating "part_of" or "belongs_to" relation by the term "of", candidate compliance checking attribute.
61. candidate compliance checking attribute, 3rd person singular present verb, negation, comparative relation, value, unit.
62. candidate subject, comparative relation, value, preposition, complementary subject.
63. negation, possessive subject restriction, comparative relation, value, unit, indicating "part_of" or "belongs_to" relation by the term "of", candidate compliance checking attribute.
64. candidate subject, preposition, comparative relation, value, unit.

Note: A pair of square brackets encloses different options that can be used in that specific slot of the pattern.