Clustering-based Approach for Building Code Computability Analysis

Ruichuan Zhang[1]; and Nora El-Gohary, A.M.ASCE[2]

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

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

Abstract

One common limitation of all automated code compliance checking methods and tools is their inability to deal with all types of building-code requirements. More research is needed to better identify the different types of requirements, in terms of their syntactic and semantic structures and complexities, to gain more insights about the capabilities and limitations of existing methods and tools (i.e., which requirements they can automatically process, represent, and/or check, and which not). To address this need, this paper proposes a new set of syntactic and semantic features and complexity and computability metrics for code computability analysis. A clustering-based approach was used to identify the different types of code sentences, in terms of their computability, using the proposed features and metrics. The approach was implemented and tested on a corpus of 6,608 sentences from the International Building Code and its amendments. The sentence clusters and identified sentence types were evaluated using intrinsic and extrinsic evaluation methods. The evaluation results indicated good clustering performance, perfect alignment between the human- and computer-identified types, and good agreement in the assignment of sentences to the types.

Keywords: Buildings; Code checking; Computability; Text analytics; Hierarchical clustering.

Introduction

Existing automated compliance checking (ACC) systems in the architecture, engineering, and construction (AEC) domain have different coverage capabilities, in terms of what type of building-code requirements they can represent, process, or check. For example, the Solibri Model Checker (SMC) (Solibri 2018), an example of commercially-available software, only covers simple, discrete, and property requirements (e.g., “risers shall be 4 inches high minimum”) and conditional requirements (e.g., “ceilings in corridors shall be not less than 2.34 meters in height”). It allows users to include or exclude building elements and adjust the properties and property values in a limited

number of rule templates. And, it requires manual effort to read the code text, identify the right rule templates to use, and enter the values of the parameters in the templates, lacking capabilities for automatically processing natural-language requirements into computable representations. In the SMARTcodes project by AEC3 and International Code Council (ICC) (AEC3 2012), an example of approaches for semi-automated requirement processing, conditional, attributive requirements are manually annotated with requirement, applicability, selection, and exception (RASE) tags (Hjelseth and Nisbet 2010), and then the annotated text is converted into the IFC-compatible XML format (Weise et al. 2017). In the SNACC system by Zhang and El-Gohary (2017b), an example of approaches for fully-automated information extraction and requirement processing, semantic modeling and natural language processing (NLP) techniques are used to automatically extract quantitative requirements (excluding qualitative requirements) from building codes into a computable format. Collectively, existing systems are limited in dealing with highly complex requirements, especially those that have hierarchically complex syntactic and semantic structures or those that require human judgement by nature (Solihin and Eastman 2015; Zhou and El-Gohary 2017; Nawari 2019).

Despite such relatively high variability in the approaches and capabilities of existing systems, there is a lack of research efforts to identify and characterize the different types of requirements or sentences in AEC regulations to better assess the coverage capabilities of existing ACC systems, identify the types of ACC systems required for different applications (Solihin and Eastman 2015), and inform future ACC efforts. For example, Solihin and Eastman (2015) identified four types of requirements based on what type of BIM data the rules require. Malsane et al. (2015) manually grouped regulations into declarative and informative clauses. These two types of classifications are useful in their intended scopes and applications, but cannot be used to assess the capabilities of existing ACC systems, in terms of what natural-language requirements they can automatically process, represent (into a computable representation), and/or check. Other research efforts that focused on developing computable representations of requirements (e.g., Eastman et al. 2009; Hjelseth and Nisbet 2010; Dimyadi and Amor 2013; Zhang and El-Gohary 2013; Dimyadi et al. 2016), by nature of their scope, did not devote efforts to analyze the different types of requirements or sentences. The knowledge of sentence types, and the syntactic and semantic features of these different types, is essential for assessing and comparing the actual and potential capabilities of ACC systems. It would help us gain insights about the capabilities and limitations of existing ACC systems (i.e., which requirements they can automatically process, represent, and/or check, and which not), in order to choose the

right system for the application at hand, improve or extend existing systems for enhanced coverage and performance, or develop new systems with increased capabilities.

To address this need, this paper first proposes a set of new features to characterize requirement sentences in building codes, in terms of their syntactic and semantic structures, to capture their syntactic and semantic complexities. This paper then proposes a set of new complexity and computability metrics. Computability aims to measure the ability of sentences to be automatically processed, represented, and checked by ACC systems based on their syntactic and semantic complexities. Sentences with complex syntactic and semantic structures are difficult to be represented in a computable representation (e.g., mvdXML), processed automatically by computational tools and techniques (e.g., NLP techniques), and checked by semi- or fully-automated ACC systems; and vice versa. A clustering-based approach was adopted to discover sentence clusters in a large corpus of building-code sentences, from IBC and its variations/amendments, using the proposed features. Sentence types were then identified based on the sentence clusters and characterized, using the proposed features and metrics, both quantitively and qualitatively.

The rest of the paper is organized as follows. The second section provides a brief review of existing ACC efforts and clustering and text mining techniques. The third section continues with an analysis of the knowledge gaps in building-code computability analysis and text clustering (in the context of our application). The fourth section presents the proposed syntactic and semantic features and complexity and computability metrics. The fifth section explains the research methodology for clustering, computability analysis, and evaluation. The sixth section presents the experimental results and discussion. Finally, the last two sections conclude the paper with contributions, conclusions, and future work.

Background

Automated Code Checking Systems

Extensive research efforts have focused on automating the process of code compliance checking in the AEC domain. Many of the existing ACC systems are semi-automated, requiring manual effort to read the code and represent the requirements in computable forms. Semi-automated efforts have used different approaches and representations. For example, Garrett and Fenves (1987) proposed a design strategy, in which standard requirements are represented as computable constraints and knowledge-base relations. Garrett and Hakim (1992) developed an object-oriented model that offers different types of representation schemes for different types of requirements. Ozkaya and Akin

(2006) proposed a design approach that uses requirement-design coupling paths to query the design products and incorporate requirements into designs for supporting code checking. More recently, query languages have been used to represent requirements and industry foundation classes (IFC)-format design information [e.g., SPARQL Query Language for RDF (Yurchyshyna and Zarli 2009), regulatory knowledge query language (Dimyadi et al. 2016)]. Visual programming languages [e.g., conceptual graphs (Solihin and Eastman 2016), Visual Code Checking Language (Preidel and Borrmann 2016)] have also been used to represent requirements and visualize them in graph-like structures. Fully-automated ACC systems, compared to semi-automated ones, aim to automate the process of extracting the information from the codes and representing the requirements in computable rule formats. For example, Zhang and El-Gohary (2013, 2015, 2016, 2017a, 2017b) and Zhou and El-Gohary (2017) proposed semantic rule-based ACC systems that use ontologies, NLP techniques, and pattern matching-based rules to extract regulatory information from the codes and represent these extracted information in the form of logic rules.

Many ACC efforts have also been led by industry bodies and government organizations, which are mostly semi-automated, such as CORENET ePlanCheck by the Singapore Building and Construction Authority, REScheck and COMcheck by the U.S. Department of Energy, and SMARTcodes by AEC3 and ICC. In most of these efforts, requirements are hard-coded by the software developers, such as in SMC, Compliance Audit Systems, Daima, Invicara, SmartReview, and UpCodes. A few other efforts rely on mechanisms such as semantic annotation to increase the system's flexibility. For example, in the AEC3 and ICC's SMARTcodes project, users first annotate the requirements with a set of semantic markups – including requirement, applicability, selection, and exception (Hjelseth and Nisbet 2010), which are then converted into mvdXML format.

Clustering for Text Analytics

Clustering is an unsupervised learning problem, which aims to find groups of similar objects in data (Aggarwal and Zhai 2012). In the context of text analytics, the clustering objects could be documents, sentences, or phrases and words. Clustering has been used in many applications such as document classification (e.g., Bekkerman et al. 2001), browsing (e.g., Cutting 2017), summarization (e.g., Yang et al. 2014), and visualization (Cadez et al. 2003). A limited number of research efforts in the construction domain have used text clustering for various purposes. For example, Ng et al. (2006) used text clustering to analyze deficiency descriptions for knowledge discovery in a facility condition assessment database. Al Qady and Kandil (2014) used a hybrid approach that included clustering

and text classification to group semantically-related project documents. And, Kifokeris and Xenidis (2017) used a notion clustering-based method to identify risk sources in technical projects.

The application of clustering algorithms can benefit building-code analytics and sentence typing for ACC from two perspectives. First, clustering is naturally suitable for finding different groups of requirements or sentences that share similar characteristics (Aggarwal and Zhai 2012, Allahyari et al. 2017). Second, clustering-based methods require minimum level of manual effort for mining the underlying patterns of the text, compared to manual methods, where the patterns are interpreted and analyzed by humans, and supervised learning-based methods, where data are manually annotated to train computational models for analyzing the patterns.

Hierarchical clustering aims to successively combine groups of data in a pairwise manner based on their pairwise similarities, until all the data are within one single group (Aggarwal and Zhai 2012). This process constructs a hierarchy of clusters, which can be intersected at a certain level to obtain a specific number of clusters. Hierarchical clustering has been widely used in the context of text analytics [e.g., Li et al. (2008), Shepitsen et al. (2008), and Lomakina et al. (2014)]. Hierarchical clustering has also been applied in solving a number of research problems in the construction domain such as assessment of social sustainability of construction projects (e.g., Valdes-Vasquez and Klotz 2012); but, to the best of the authors' knowledge, no efforts focused on using hierarchical clustering for document analysis.

State of the Art and Knowledge Gaps

State of the art and knowledge gaps in **building-code computability and typing analysis**: A very limited number of research efforts have been undertaken to formally characterize the different types of requirements or sentences in building codes. Solihin and Eastman (2015) grouped requirements/rules based on what type of BIM data the rules require, and accordingly identified four general classes of rules: (1) rules that only need a small number of explicit data that exist in the BIM dataset; (2) rules that need simple derived attribute values; (3) rules that need extended data structures; and (4) rules that need a "proof of solution" (e.g., example cases/sentences and/or manual hypothetical checking processes). This effort is very useful in assessing the existence of BIM data for checking the requirements, but, by nature of its scope, does not address the classification of requirements by their computability (e.g., the ability of ACC systems to automatically process natural-language requirements into computable representations). Malsane et al. (2015) grouped the clauses in the England and Wales Fire Safety Building

Regulations (EWFSBR) Part B1 into two types: (1) declarative clauses: clauses that have obviously checkable information and thus are computer interpretable (e.g., simple geometrical rules); and (2) informative clauses: clauses that have information that is not obviously checkable or needs human interpretation and thus are not computer interpretable. This type of classification could be useful as a first-level binary (black-and-white) classification, but cannot be used for the purpose of computability analysis, because it does not identify the different subtypes of computer-interpretable requirements – some subtypes would be computer-interpretable by some systems but not the others. Also, the aforementioned criteria for classifying a clause into the first category versus the second requires human interpretation, lacking well-defined text features that could support the automation of the classification and the computability analysis process.

State of the art and knowledge gaps in **text clustering**: In the computational linguistics domain, text clustering efforts have mainly focused on document and phrase/word clustering, both separately (e.g., topic model-based document clustering, WordNet-based phrase/word clustering) and simultaneously [e.g., word clustering and coclustering (Kilicoglu and Bergler 2009)], but rarely studied sentence clustering. Compared to document clustering, sentence clustering usually has a sparser feature space because a sentence has significantly fewer words than a document and thus requires effective feature selection. In terms of the purpose of clustering, existing sentence and/or text clustering efforts evaluated either the semantic similarity (Naughton et al. 2010, Fodeh et al. 2011, Yang et al. 2014) or the syntactic similarity of the text only (Massung et al. 2013), without combining both analyses in the clustering problem.

State of the art and knowledge gaps in **features and metrics for text analytics**: Various features and metrics have been proposed or adopted for supporting text analytics. These features mainly included (1) syntactic features, such as part-of-speech (POS) tag frequency and parsing tree structural features (e.g., Massung et al. 2013), or (2) semantic features, such as bag of words, word frequency, and features based on WordNet (e.g., Naughton et al. 2010, Fodeh et al. 2011) and document graph models (e.g., Yang et al. 2014). These features are effective in text analytics tasks such as text categorization (e.g., Massung et al. 2013) or topic-based text clustering (e.g., Naughton et al. 2010, Yang et al. 2014). However, their effectiveness is limited in requirement computability analysis because: (1) they are mostly lexical- or word-level features and cannot directly reflect requirement- and sentence-level complexity; and (2) they lack the ability to capture the semantic characteristics specific to building-code text such as

the essential and secondary semantic information elements. Some of these features were further used in the literature for calculating or deriving text complexity metrics. Existing metrics for text complexity analysis can be grouped into three categories: (1) lexical complexity metrics, such as number of infrequent words; (2) syntactic complexity metrics, such as the length of sentences, number of infrequent structural features, and number of constituents in the parsing trees (e.g., Ambati et al. 2016); and (3) semantic complexity metrics, such as knowledge graph-based measures like node degree, length of shortest path, and number of connected components (e.g., Štajner and Hulpuş 2018). These metrics are useful for text complexity analysis in terms of readability (Ambati et al. 2016, Štajner and Hulpuş 2018) but cannot be used to analyze text computability, because they are based on the comprehensibility of the text by human readers, rather than the ability of the text to be automatically processed, represented, and checked by computational systems (e.g., ACC systems). Also, similar to existing features for text analytics, these metrics are limited in indicating the syntactic and semantic complexity of building-code requirements, because they do not capture the syntactic and semantic characteristics that are specific to building-code text (Zhou and El-Gohary 2017).

Proposed Features and Metrics for Building-Code Computability Analysis and Typing

Features for Building-Code Computability Analysis and Typing

Syntactic Features

Four syntactic features are proposed for representing the complexity of the syntactic structures of building-code requirement sentences. The features are capturing the syntactic lengths of the sentence fragments (i.e., phrases, clauses, or sentences) and the syntactic heights of the constituency trees corresponding to the sentence fragments, as shown in Fig. 1. The syntactic lengths and heights include: (1) the lengths of the sentence fragments for each fragment labeled with the following phrase- and clause-level tags in the sentence-level phrase structure rules: NP (noun phrase), VP (verb phrase), PP (preposition phrase), and SBAR (clause introduced by a subordinating conjunction such as that, where, and when); (2) the length of the whole sentence; (3) the heights of the constituency trees for each fragment labeled with NP, VP, PP, or SBAR; and (4) the height of the entire constituency tree. These features were chosen for two reasons. First, the constituency tree consists of nested linguistic constituents that represent the syntactic structure of a sentence fragment (Jurafsky and Martin 2014). The higher the constituency tree, the more nested the syntactic structures of the sentence fragment. Second, the longer the sentence and sentence

fragments, the more syntactic information contained in the sentence fragment. The more nested the structure of the sentence fragment and the more information it contains, the more complex the fragment.

The four features are: (1) complexity of NP; (2) complexity of VP; (3) complexity of PP or SBAR, whichever is higher (because both have similar functions, i.e., to represent modification or adjunct meanings of the sentence); and (4) complexity of the whole sentence. The feature values are computed as per Eqs. (1) to (3), where SPC is NP, VP, PP, or SBAR; l_i is the length of the phrase or clause i; h_j is the height of the constituency tree of the phrase or clause j; F is the set of all the phrases or clauses labeled with SPC; L is the length of the whole sentence S; and H is the height of the constituency tree of the whole sentence.

$$Complexity\ of\ SPC\ = \sqrt{\max_{i \in F} l_i \times \max_{j \in F} h_j} \quad (1)$$

$$Complexity\ of\ PP/SBAR = \max(Complexity\ of\ PP, Complexity\ of\ SBAR) \quad (2)$$

$$Complexity\ of\ S = \sqrt{L \times H} \quad (3)$$

Semantic Features

Four semantic features are proposed for representing the complexity of the semantic structures of building-code requirement sentences. Three steps were used for feature identification: feature analysis, selection, and synthesis. First, the essential and secondary semantic information element (SIE) features were analyzed. Table 1 shows all the semantic information elements covered in this analysis. Essential SIEs are usually necessary for defining a quantitative requirement, such as subject, compliance checking attribute, deontic operator indicator, comparative relation, quantitative relation, quantity value, and quantity unit (Zhang and El-Gohary 2015). The more essential SIEs in a sentence, the higher the computability of the sentence, and vice versa. Secondary SIEs are not essential but may exist in defining a quantitative or qualitative requirement, usually adding complexity to the sentence, such as restrictions (e.g., a subject restriction places a constraint on the definition of the subject) and references. The less secondary SIEs in a sentence, the higher the computability of the sentence, and vice versa. Second, low-variance features were removed because they are non-discriminative. The feature analysis showed that most of the requirement sentences (i.e., over 95%) have subjects and deontic operator indicators; and, thus, these two features were removed. Third, high-covariance features were synthesized into one feature to improve the sentence type characterization. Based on the feature analysis, the following four high co-variance features were synthesized into one (called quantitative semantic information): comparative relation, quantitative relation, quantity value, and

quantity unit. Accordingly, the final four semantic features are: presence (binary value) of compliance checking attribute, quantitative semantic information, restriction, and reference.

Metrics for Building-Code Computability Analysis and Typing

Syntactic Complexity

The syntactic complexity metric aims to measure the complexity of the syntactic structures of sentences in a cluster of requirement sentences, in terms of the aforementioned syntactic features. The syntactic complexity is computed based on the average feature values of the complexity of NP, VP, PP/SBAR, and S (denoted as $\overline{NP}$, $\overline{VP}$, $\overline{PP/SBAR}$, and $\bar{S}$), as per Eqs. (4) and (5), where SCC is the syntactic cluster complexity before normalization, and minSCC and maxSCC are the minimum and maximum SCCs among all the clusters, respectively. Eq. (4) assigns more weight to the complexity of the whole sentence (i.e., the S feature). The complexity ranges from 0 to 1, where 0 represents minimum syntactic complexity and 1 represents maximum syntactic complexity. The lower the syntactic complexity of a cluster, the less complex the syntactic structures of the sentences in the cluster, and vice versa.

$$\text{Syntactic cluster complexity (before normalization)} = \text{SCC} = \frac{1}{2}\left(\frac{\overline{NP} + \overline{VP} + \overline{PP/SBAR}}{3} + \bar{S}\right) \quad (4)$$

$$\text{Syntactic cluster complexity} = \frac{SCC - minSCC}{maxSCC - minSCC} \quad (5)$$

Semantic Complexity

The semantic complexity metric aims to measure the complexity of the semantic structures of sentences in a cluster of requirement sentences, in terms of the aforementioned semantic features. The semantic complexity of a corpus is impacted by two factors, as per Eq. (6): the essential semantic information (ESI) absence factor and the secondary semantic information (SSI) presence factor. The absence of ESI (i.e., compliance checking attribute or quantitative semantic information) and the presence of SSI (i.e., restrictions or references) increase semantic complexity and decrease computability, as per Eq. (6). The ESI absence factor is calculated as per Eq. (7), where *E* is the percentage of sentences in the corpus that have missing ESI. The SSI presence factor is calculated as per Eq. (8), where *RS* is the percentage of sentences in the cluster that have restrictions and *RF* is the percentage of sentences in the cluster that have references. Eq. (6) assigns more weight to the SSI presence factor, because the presence of SSI creates more semantic complexity than the absence of ESI. Eq. (8) similarly assigns more weight to restrictions because they create more complexity compared to references. Semantic cluster complexity ranges from 0 to 1, where 0

represents minimum semantic complexity (i.e., all sentences in the cluster have no missing ESI and no SSI) and 1 represents maximum semantic complexity (i.e., all sentences in the cluster have both types of SSI – restrictions and references – and miss at least one type of ESI). The semantic feature analysis was performed using the information extraction and transformation rules (Zhang and El-Gohary 2013). The lower the semantic complexity of a cluster, the less complex the semantic structures of the sentences in the cluster, and vice versa.

$$\text{Semantic cluster complexity} = \frac{1}{3} \times ESI\ absence\ factor + \frac{2}{3} \times SSI\ presence\ factor \quad (6)$$

$$\text{ESI absence factor} = E \quad (7)$$

$$\text{SSI presence factor} = \frac{4}{7} \times RS + \frac{3}{7} \times RF \quad (8)$$

Computability

The computability metric aims to measure the ability of requirement sentences to be represented, processed, and checked by ACC methods and systems based on their syntactic and semantic structures. The computability was computed based on the syntactic and semantic complexity metrics, as per Eq. (9), where computability is 1 minus the average of the two metrics. It ranges from 0 to 1, where 0 represents minimum computability and 1 represents maximum computability.

$$\text{Cluster computabiliy} = 1 - \left(\frac{1}{2} \times Syntactic\ cluster\ complexity + \frac{1}{2} \times Semantic\ cluster\ complexity\right) \quad (9)$$

Methodology for Clustering-based Building-Code Computability Analysis and Typing

A methodology for analyzing the computability of building codes and identifying the types of sentences, using the proposed features and metrics, was proposed and implemented on a corpus of sentences. The methodology included five primary steps, as illustrated in Fig. 2: data preprocessing, first-level clustering, subclustering, computability analysis and sentence-typing, and evaluation.

Data Preprocessing

A total of 6,608 sentences were randomly selected from the IBC 2009 and the 2015 IBC Amendment of the City of Champaign. Three steps were conducted for data preparation. First, different formats (e.g., PDF and HTML) were converted into TXT format. Second, the text was split into sentences and non-sentence fragments (e.g., requirement titles) based on punctuations and requirement indices, using the Natural Language Processing Tool Kit (NLTK) (Bird et al. 2009), and the non-sentence fragments were removed. Third, the special symbols were removed. Three

NLP techniques were used for data preprocessing: tokenization, POS tagging, and constituency parsing. Tokenization aims to split a sentence into units (e.g., words and punctuations). POS tagging and constituency parsing aim to analyze the syntax of a sentence in multiple levels (e.g., sentence, phrase, and word levels) (Jurafsky and Martin 2014). The NLTK was used for tokenization. The Stanford CoreNLP (Manning et al. 2014), built in python, was used for POS tagging and parsing. The tokenized text, and the corresponding POS tags and constituency parsing trees, were further used to generate the syntactic and semantic features.

First-level Clustering

First-level clustering aimed to cluster the sentences according to their syntactic and semantic-structure similarities, using the proposed syntactic and semantic features. The hierarchical clustering consisted of three steps: distance calculation, hierarchical clustering, and determining the number of clusters. First, the pairwise Euclidean distances of the sentences were calculated using the feature values, in order to measure the syntactic and semantic similarities between pairs of sentences. Second, different methods for hierarchical clustering analysis were tested and compared, including simple, complete, average, McQuitty, median, centroid, and Ward's (Aggarwal and Zhai 2012). They were also compared with two other commonly used distance-based clustering methods as baselines: k-means (Arthur and Vassilvitskii 2006) and partition around medians (PAM) (Kaufman and Rousseeuw 2009). Third, to determine the optimal number of clusters, the elbow rule was adopted, which means that a specific number of clusters is optimal when the addition of more clusters results in only marginal improvement – which can be seen as an elbow-like shape in the plot of percentage of variance explained by the clusters (Ketchen and Shook 1996).

Subclustering

Subclustering aimed to further decompose the clusters that have low average silhouette coefficients into granular clusters that are different from each other in terms of their constituent-level features such as the frequencies of phrase structure rules and POS tags. This is because the low coefficient values indicate that the clusters are not well represented by the features used in the first-level clustering and thus need additional features to better characterize them for the purpose of sentence typing. The following constituent-level features were used: (1) the frequency of each POS tag; (2) the frequency of each bigram of POS tags, which are pairs of two consecutive POS tags; and (3) the frequency of each phrase structure rule. To penalize the common features (which are not discriminative across different clusters) and to promote the discriminative features (which contribute to clustering), the raw features were

297 weighted using the TF–IDF weighting scheme. Eqs. (10) and (11) (Salton and Buckley 1988) were used, where f is
298 the frequency of bigrams of POS tags or phrase structure rules, N is the total number of sentences, and n is the
299 number of sentences that contain a specific feature.

300 $$\text{Augmented normalized term frequency} = 0.5 + 0.5\frac{f}{maxf} \quad (10)$$

301 $$\text{Inverse document frequency} = log\frac{N}{n} \quad (11)$$

302 Two feature analyses were conducted empirically to identify the discriminative features, including leave-one-out
303 feature analysis and feature combination analysis. The local learning algorithm by Yao et al. (2015) was used for
304 feature selection to deal with high dimensionality for finding meaningful clusters. This algorithm aims to find a
305 subset of features that makes the sum of the distances between each datum and its nearest datum small, and
306 maximizes the sum of the average distances of each datum to all other data (Yao et al. 2015). The output of the
307 algorithm – the weights of features – was used to compute the weighted feature values. The hierarchical clustering
308 steps were similar to those used in first-level clustering.

309 ***Cluster Computability Analysis and Sentence Typing***

310 Three steps were conducted to analyze the computability of the sentence clusters: cluster representation, analysis of
311 the syntactic and semantic complexities and computability of the clusters, and cluster characterization. First, each
312 cluster was represented by the average feature values (the four syntactic and four semantic features) of the sentences
313 in the clusters. Second, the complexity and computability metrics of the clusters were computed using the average
314 feature values [as per Eqs. (4) to (9)]. Third, the clusters were characterized to better describe and compare the
315 clusters – and thus the sentence types. The average feature values, syntactic and semantic complexity values, and
316 cluster computability values were discretized based on their quartile and median values (Dougherty 1995).
317 Discretization groups a number of continuous values into a smaller number of "bins". A scale of 1 to 6 was then
318 used to represent the level of complexity (very simple to very complex) and computability (very low to very high),
319 as per Table 2. Subclusters were first characterized in the same way as their parent clusters, then additionally
320 characterized using their constituent-level features. The level of computability of the children clusters was adjusted
321 accordingly.

Sentence types were then identified and characterized based on the clusters – assuming each cluster (or subcluster) has only one sentence type. Each sentence type was described with a name, a description, and an example sentence. Three steps were conducted to identify and characterize the sentence types: (1) for each cluster, a name that reflects the characteristics of the sentence type was selected. For example, a type with high syntactic cluster complexity, one or more restrictions, one or more references, and no quantitative semantic information, was named "complex, qualitative, restricted, and with references"; (2) each identified sentence type was briefly described in terms of its syntactic and semantic characterization, computability level, and relation to other types, if any; and (3) for each cluster, an example sentence was selected to illustrate the sentence type and its characteristics.

Evaluation of Clustering and Sentence Types

Intrinsic Evaluation

Intrinsic evaluation aims to evaluate the sentence clusters by calculating and comparing the quality metrics that describe intra-cluster and inter-cluster similarities (Manning et al. 2008). Two intrinsic metrics were used for both first-level clustering and subclustering: average silhouette coefficient and cophenetic coefficient. The average silhouette coefficient (Rousseeuw 1987) was used to measure the performance of all clustering algorithms. It is the average of the silhouette coefficients of all the sentences in a dataset or a cluster. The silhouette coefficient $s(i)$ of a sentence i is defined as per Eq. (12), where $a(i)$ is the average difference between sentence i and the other sentences in the same cluster and $b(i)$ is the lowest average difference between i and the other clusters. The coefficient ranges from -1 to 1, where a value near 1 indicates that the sentence is far from the neighboring clusters, 0 indicates that the sentence lies on the boundary between two or more clusters, and a negative value indicates that the sentence might be assigned to a wrong cluster. A coefficient between 0 and 0.2 indicates poor clustering, between 0.2 and 0.5 fair clustering, and higher than 0.5 good clustering (Sarstedt and Mooi 2014). When all the sentences in a cluster have significantly below-average silhouette coefficients, this indicates that the cluster might need to be further subclustered.

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}} \quad (12)$$

The cophenetic coefficient (Sneath and Sokal 1973) was calculated to justify the choice of the hierarchical clustering method. It shows how well a dendrogram generated by the hierarchical clustering process preserves the pairwise distances of the original data. It is defined as per Eq. (13), where Y_{ij} is the difference between sentences i and j, Z_{ij}

is the height of the node in the dendrogram at which sentences i and j are first combined into one group, $\bar{y}$ is the average of all Y_{ij}, and $\bar{z}$ is the average of all Z_{ij}. The cophenetic coefficient ranges from 0 to 1, with 1 indicating that the dendrogram perfectly reflects the pairwise distances of the original dataset and thus the dataset is suitable for hierarchical clustering, and 0 indicating the opposite (Sneath and Sokal 1973).

$$c = \frac{\sum_{i<j}(Y_{ij} - \bar{y})(Z_{ij} - \bar{z})}{\sqrt{\sum_{i<j}(Y_{ij} - \bar{y})^2 \sum_{i<j}(Z_{ij} - \bar{z})^2}} \quad (13)$$

Extrinsic Evaluation

Extrinsic evaluation, here, aims to evaluate the identified sentence types using human expert judgement (Manning et al. 2008). Five participants – the first author and four experts including two from academia (faculty) and two from industry – conducted the extrinsic evaluation. The participants manually identified building-code sentence types from a set of testing sentences and assigned the testing sentences to these types; and the human-generated sentence types and assignments were compared to the computer-generated types and assignments. Purposive sampling strategy was adopted for selecting the participating experts. Purposive sampling aims to pinpoint a specific type of participants according to predefined selection criteria (Clark and Creswell 2008). Three main selection criteria were defined: (1) expertise in the AEC domain; (2) familiarity with building codes and compliance checking processes; and (3) awareness of natural language processing and text analytics techniques. The authors used purposive sampling because (1) it is suitable for small specialized populations (e.g., experts) (Etikan et al. 2016); and (2) it enables obtaining information from a concentrated, carefully selected sample (Clark and Creswell 2008). Expert evaluation of knowledge discovery processes has been commonly conducted with a small purposively sampled set of participants [e.g., seven (El-Diraby and Osman 2011), six (Salama and El-Gohary 2013), five (Jin 2010), and four (Alashwal and Abdul-Rahman 2014)].

The evaluation process consisted of three stages: testing dataset preparation, testing sentence typing, and human-computer agreement assessment. To develop a testing dataset, a sample of 160 sentences were randomly selected from the 6,608 sentences. The sentences were sampled from the whole dataset (excluding the example sentences selected to illustrate the identified sentence types) following a stratified sampling strategy (Särndal et al. 2003), i.e., the sentences were sampled from each of the clusters, where the sample sizes are proportional to the cluster sizes.

The manual typing aimed to group testing sentences using two steps: identifying building-code sentence types (based on computability and syntactic and semantic features) and assigning testing sentences to these types. Detailed guidelines for the typing were provided and explained, including (1) the definition of computability and the set of syntactic and semantic features; (2) example building-code sentences, with their computability and syntactic and semantic features explained; and (3) a template of how a type should be described (e.g., very simple, quantitative, unrestricted, with no references). The participants repeated the sentence type identification and assignment steps until the following two terminating criteria were met: (1) each testing sentence has been assigned to a human-identified type; and (2) no new types are identified.

The agreement between the human- and computer-identified types was assessed using Jaccard Index (Agresti 2003), as per Eq. (14), where H is the collection of types identified by the human evaluators and C is the collection of types identified by the computer. A Jaccard Index of 1 indicates that the human evaluators and the computer identify the same set of types. The agreement between the human- and computer-assignments of sentences to the types was assessed using percentage of agreement (Hallgren 2012). The percentage of agreement is defined as the percent value of the ratio of the number of testing sentences that were assigned to the same type by the human evaluators and the computer, to the total number of testing sentences. A percentage of agreement close to 1 indicates that there is a very good alignment between the human- and computer-assignment of sentences (Stemler 2004). Fig. 3 illustrates the entire extrinsic evaluation process.

$$\text{Jaccard} = \frac{|H \cap C|}{|H \cup C|} \quad (14)$$

Application-oriented Evaluation

This type of extrinsic evaluation aims to further evaluate the identified sentence types using an application-oriented way. The evaluation method uses a testing dataset (set of building-code sentences) in an ACC application (here regulatory information extraction) and then assess if the different types of sentences will be associated with different levels of computability. Regulatory information extraction aims to automatically extract computable information from natural-language building codes (Zhang and El-Gohary 2013, Li et al. 2016, Zhou and El-Gohary 2017, Zhong et al. 2020). The hypothesis is that sentence types with high computability will have significantly higher information extraction performance than those types with low computability (using existing ACC systems which are currently still limited in dealing with complex types).

The evaluation consisted of three stages: testing dataset preparation, information extraction from testing sentences, and information extraction performance assessment. To develop a testing dataset (different from the one used in prior evaluation), a sample of 60 sentences were selected from the 6,608 sentences; five sentences were randomly sampled from each of the twelve sentence types. The machine learning-based method by Zhang and El-Gohary (2020) was used for information extraction. Accordingly, the nine types of SIEs (see Table 1) were extracted from the testing sentences. To assess the information extraction performance, three metrics were used: precision, recall, and F1-measure, as shown in Eqs. (15) to (17), where for a specific type of SIE, *TP* is the number of true positives (i.e., number of SIE instances correctly extracted), *FP* is the number of false positives (i.e., number of SIE instances incorrectly extracted), and *FN* is the number of false negatives (i.e., number of SIE instances not extracted but should have been) (Zhai and Massung 2016).

$$\text{Precision} = \frac{TP}{TP + FP} \quad (15)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (16)$$

$$\text{F1-measure} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (17)$$

Experimental Results and Discussion

Results of Cluster Computability Analysis and Sentence Types

Table 3 summarizes the characteristics and computability analysis of all the clusters that resulted from the first-level clustering, including the syntactic features (i.e., complexity of NP, VP, PP/SBAR, and S), semantic features (i.e., presence of compliance checking attribute, quantitative semantic information, reference, and restriction), syntactic and semantic cluster complexity metrics, and cluster computability metric. Table 4 shows the features of the children clusters that resulted from the subclustering, the feature interpretations, and the computability comparison to their sibling clusters.

Based on the clusters, a total of twelve types of sentences were identified. The following list shows the different types of sentences and their descriptions, which were identified based on the cluster characterization and computability analysis. Table 5 shows an example sentence for each type, as well as the sentence type frequencies (i.e., percentages of the sentence types in the experimental dataset) and their computability levels. Sentences that belong to the same type have similar – but not identical – levels of syntactic and semantic complexities and

computability. For example, although sentences that belong to the same type share similar NP, VP, PP/SBAR, and S complexities and similar semantic information elements, they are likely to have different sentence lengths. The identified types and their computability levels should thus be taken as a guide, and not as an exact measure. Sentence types that are more similar to each other in terms of their syntactic and semantic features compared to other types such as those that belong to the same parent cluster, were numbered as sibling types (e.g., Type 1.1 and Type 1.2).

- Type 1.1 (very simple, quantitative, unrestricted, with no references): The sentences have very simple structures, with simple NP, VP, and very simple PP/SBAR. They have no missing ESI (i.e., compliance checking attribute and quantitative semantic information), and have no SSI (i.e., restriction and reference). They cover quantitative requirements. They can be represented/coded in many of the existing ACC systems. Type 1.1 corresponds to Cluster 1.1.
- Type 1.2 (simple, quantitative, unrestricted, with no references): Similar to Type 1.1. But, compared to Type 1.1 sentences, Type 1.2 sentences are more complex syntactically (e.g., including more preposition phrases and/or conjunctions) and semantically (e.g., including multiple requirements). Type 1.2 corresponds to Cluster 1.2.
- Type 2 (very simple, qualitative, unrestricted, with no references): The sentences have very simple structures, with very simple NP, VP and PP/SBAR. They have missing ESI, from both types (i.e., compliance checking attribute and quantitative semantic information), but have no SSI (i.e., restriction and reference). They cover qualitative requirements. Type 2 corresponds to Cluster 2.
- Type 3.1 (simple, descriptive, unrestricted, with references): The sentences have simple structures, with very simple NP and PP/SBAR, and simple VP. They have missing ESI, from both types (i.e., compliance checking attribute and quantitative semantic information), and have one type of SSI (i.e., reference). Most of the sentences function as general descriptions or references to other sections/codes. Type 3.1 corresponds to Cluster 3.
- Type 3.2 (moderately simple, descriptive, unrestricted, with references): The sentences have moderately simple structures, with very simple PP/SBAR, and moderately simple NP and VP. They have one type of ESI (i.e., quantitative semantic information) missing, and have one type of SSI (i.e., reference). Most of the sentences function similar to Type 3.1 sentences, as general descriptions or references to other sections/codes, but are more complex both syntactically and semantically. Type 3.2 corresponds to Cluster 4.

- Type 4.1 (moderately complex, quantitative, restricted, with no references): The sentences have moderately complex to complex structures, with moderately complex NP, and complex VP and PP/SBAR. The sentences are likely to have one or more clauses of different types (e.g., SBAR). The sentences have no missing ESI (i.e., compliance checking attribute and quantitative semantic information), but have one type of SSI (i.e., restriction), which might be in the VP and/or PP/SBAR. Most of the sentences are quantitative and function similar to Type 1 sentences, but are much more complex both syntactically and semantically, mostly due to the restrictions. Type 4.1 corresponds to Cluster 5.1.
- Type 4.2 (complex, quantitative, restricted, with no references): Similar to Type 4.1. But, compared to Type 4.1 sentences, Type 4.2 sentences are more complex syntactically (e.g., including more noun phrases) and semantically (e.g., including both restrictions and multiple requirements). Type 4.2 corresponds to Cluster 5.2.
- Type 5.1 (moderately complex, qualitative, restricted, with no references): The sentences have moderately complex structures, with simple PP/SBAR, moderately simple NP, but moderately complex VP. They have missing ESI, from both types (i.e., compliance checking attribute and quantitative semantic information), and have one type of SSI (i.e., restriction), which might be in the VP and/or PP/SBAR. Most of the sentences are qualitative and function similar to Type 2 sentences, but are more complex both syntactically and semantically, mostly due to restrictions. Type 5.1 corresponds to Cluster 6.
- Type 5.2 (moderately complex, descriptive, restricted, with references): The sentences have moderately complex structures, with simple PP/SBAR, but moderately complex NP and VP. They have missing ESI, from both types (i.e., compliance checking attribute and quantitative semantic information), and have both types of SSI (i.e., reference and restriction), which might be in the NP, VP, and/or PP/SBAR. Most of the sentences function similar to Type 3.2 sentences, as descriptions/definitions, but are more complex both syntactically and semantically, mostly due to restrictions. Most of the sentences are syntactically similar to those in Type 5.1, but include references. Type 5.2 corresponds to Cluster 7.
- Type 6.1 (complex, quantitative, restricted, with references): The sentences have very complex structures, with complex NP and very complex VP and PP/SBAR. They are likely to have one or more clauses of different types (e.g., SBAR). They have no missing ESI (i.e., compliance checking attribute and quantitative semantic information), but have both types of SSI (i.e., reference and restriction), which might be in the NP and

PP/SBAR. Most of the sentences are quantitative and function similar to Type 4 sentences, but are more complex both syntactically and semantically. Type 6.1 corresponds to Cluster 8.1.

- Type 6.2 (very complex, quantitative, restricted, with references): Similar to Type 6.1. But, compared to Type 6.1 sentences, Type 6.2 sentences are more complex syntactically (e.g., including long and nested noun phrases) and semantically (e.g., using complex, or even ambiguous, descriptions). Type 6.2 corresponds to Cluster 8.2.
- Type 7 (complex, descriptive, restricted, with references): The sentences have complex structures, with moderately complex VP, and very complex NP and PP/SBAR. The sentences are likely to have one or more clauses of different types (e.g., SBAR). They have one type of ESI (i.e., quantitative semantic information) missing, and have both types of SSI (i.e., restriction and reference), which might be in the NP, VP, and/or PP/SBAR. Most of the sentences function similar to Type 5.2 sentences, as descriptions/definitions and references to other sections/codes, but are more complex syntactically. Type 7 corresponds to Cluster 9.

The identified sentence types can be further used to characterize existing ACC methods/systems, in terms of which sentence types they can process and/or represent. For example, ACC methods/systems that can represent sentences of Types 1.1 and 1.2 and some of the sentences of Types 4.1 and 4.2 (e.g., Solibri Model Checker) are able to cover sentences (1) with very simple to moderately complex structures, and (2) with no missing ESI and with no or simple SSI. These methods/systems have medium levels of coverage capabilities because the types they can represent have a maximum level of medium computability. ACC methods/systems that can represent sentences of Types 1.1, 1.2, 4.1, 4.2, and 6.1 and 6.2 [e.g., the SNACC system (Zhang and El-Gohary 2017b)] are able to cover sentences (1) with very complex structures, and (2) with no missing ESI, but with SSI. These methods/systems have higher levels of coverage capabilities because the types they can represent range from very high to very low level of computability.

Results of Intrinsic Evaluation

Table 6 summarizes the performance results for the tested clustering algorithms. For first-level clustering, the Ward's hierarchical clustering analysis method performed the best in terms of average silhouette coefficient, and was therefore selected. Fig. 4 shows the plot of the percentage of variance explained by the clusters. Based on the plot, nine was chosen as the optimal number of clusters according to the elbow rule. The average silhouette coefficient is 0.742, which indicates good clustering performance. The cophenetic coefficient of the whole dataset is

0.852, which indicates that the dendrogram that is generated by the hierarchical clustering algorithm reflects the pairwise distances of the sentences well. According to the silhouette coefficient plot (Fig. 5), the coefficients of Clusters 5 and 8 are lower than the coefficient of the whole dataset; and, thus, the two clusters were subclustered. Cluster 1 was additionally subclustered for the purpose of finding the simplest quantitative sentence type, which can be represented and processed by all ACC systems.

For subclustering, the centroid and McQuitty methods performed the best for Clusters 5 and 8, respectively; and the complete, McQuitty, and Ward's methods, as well as the PAM, performed equally well for Cluster 1. It is likely that multiple clustering algorithms were optimal because of the small size of Cluster 1. The average silhouette coefficients of Clusters 1, 5, and 8 after subclustering are 0.674, 0.661, and 0.515, respectively, indicating good performance. The cophenetic coefficients of Clusters 1, 5, and 8 after subclustering are 0.747, 0.809, and 0.724, respectively, indicating that the dendrograms generated by the hierarchical clustering algorithms reflect the pairwise distances of the sentences in each of the three clusters well.

Feature selection and synthesis improved the clustering performance, as shown in Table 7. For first-level clustering, feature selection and synthesis increased the average silhouette coefficient from 0.183 to 0.742, compared to the initial features. For subclustering, TF-IDF weighting and the local learning feature selection and weighting increased the average silhouette coefficient by an average of 0.376.

Results of Extrinsic Evaluation

The external evaluation results showed a Jaccard Index of 1, which indicates perfect alignment between the human- and computer-identified types. The results also showed a percentage of agreement of 80%, which indicates good alignment between the human- and computer-assignments of sentences to the types (Stemler 2004). An analysis of the results showed two main sources of disagreement in the assignments. First, like features used in any other text analytics or clustering task, the features used in the sentence clustering contained errors. No existing NLP algorithm/tool can achieve 100% performance, especially for relatively complex tasks such as constituency parsing. For example, PP is subject to attachment errors in constituency parsing (Jurafsky and Martin 2014), mistaking the sentence-level phrase structure rule "S→NP VP PP" as "S→NP VP" and thus causing wrong feature values for complexity of PP. Errors in constituency parsing may also cause errors in SIE extraction, and thus lead to wrong

semantic feature values. Second, some sentences that are far from the cluster centers, lying on the boundary between two or more adjacent clusters, were misclustered.

Results of Application-oriented Evaluation

Table 8 summarizes the results of the application-oriented evaluation. As shown, sentence types with very high or high computability (i.e., Types 1.1, 1.2, and 2) achieved relatively higher information extraction performance (i.e., >= 95% precision, recall, and F1-measure) than types with low or very low computability (i.e., Types 6.1, 6.2, and 7, which achieved <= 80% precision, recall, and F1-measure). This proves the stated hypothesis (see "Application-oriented Evaluation" subsection).

Limitations

Two limitations of the experiments are acknowledged. First, the testing corpus, although large in scale, is based on only IBC and IBC variations/amendments. In future work, the authors plan to analyze the computability of other building codes and standards that have been covered by existing ACC systems (e.g., International Energy Conservation Code) using the proposed features and metrics and following the clustering-based approach. Second, in the application-oriented evaluation, the proposed features and metrics, and the identified types, were only tested in one ACC task (i.e., regulatory information extraction) and using one method/tool. In future work, the authors plan to test the features and metrics in other ACC tasks (e.g., rule representation) and using different methods/tools (e.g., mvdXML, visual programming languages, and query languages).

Contribution to the Body of Knowledge

This paper contributes to the body of knowledge on three main levels. From the perspective of building-code analytics, first, the paper proposes a set of features that capture the syntactic and semantic structural complexity of requirement sentences. Second, it proposes a number of complexity and computability metrics to support the analysis of code computability. Third, it uses clustering to identify and characterize the different types of requirements based on these features and metrics. The experimental results show that the proposed features and metrics, along with the clustering-based methodology, were able to support the discovery of sentence types, the analysis of their characteristics, and the assessment of their computability. The proposed features and metrics could also be used to assess and compare the capabilities of different ACC systems and methods in a measurable and consistent manner.

Second, from a practical perspective, the paper identifies different types of sentences in the building code, in terms of their syntactic and semantic features and levels of computability. The knowledge of such sentence types could help guide the use and development of ACC methods/systems in three directions. First, it could help us better characterize existing ACC methods/systems, in terms of which sentence types they can process and/or represent. Second, it could help users select the right ACC system for the application at hand. Third, it could help us better understand the different types of sentences in the code and their characteristics as we embark on developing new ACC methods/systems or extending the capabilities of existing ones. For example, it could help us set the research agenda for smart code analytics and ACC, starting from the less complex and more computable sentences and moving up to the more complex and less computable ones.

Third, from a clustering perspective, this paper provides a comparison of different clustering algorithms and analysis methods, in the context of building-code analytics and sentence-type identification. The experiment results show that hierarchical clustering algorithms outperformed other distance-based clustering algorithms. The best method for hierarchical clustering analysis varied for different clustering problems. For example, for first-level clustering, the Ward's method outperformed the other methods.

Conclusions and Future Work

In this paper, a clustering-based approach for building-code computability analysis was used. A set of computability features, including both syntactic and semantic features, were proposed and used for characterizing and clustering the code sentences. The syntactic features represent the complexity of the syntactic structure of a sentence by capturing the syntactic lengths of the phrases, clauses, and sentence and the syntactic heights of the constituency trees corresponding to the sentence fragments. The semantic features are features indicating the semantic meaning, content, structure, and complexity of a requirement, including presence of compliance checking attribute, quantitative requirement descriptions, restrictions on concepts/requirements, references to other sections/codes, etc. A number of computability metrics were also proposed to analyze the syntactic and semantic complexities and computability of the sentence clusters: syntactic cluster complexity, semantic cluster complexity, essential semantic information (ESI) absence factor, secondary semantic information (SSI) presence factor, and cluster computability. Clusters with low average silhouette coefficients were subclustered using additional word-level features such as phrase structure rules and POS tags. The sentence clusters were evaluated intrinsically, based on the average

589 silhouette coefficient and cophenetic coefficient; and the identified sentence types were evaluated extrinsically,
590 using both expert evaluation and application-oriented evaluation.

591 A total of 6,608 sentences from IBC 2009 and Champaign IBC amendment 2015 were analyzed using the proposed
592 approach. A total of twelve types and subtypes of sentences were identified, characterized in terms of their
593 computability features and metrics, and classified into six complexity levels (very low to very high complexity) and
594 computability levels (very high to very low computability). For example, Type 1.1, which has the lowest complexity
595 and highest computability, has simple NP and VP, very simple PP/SBAR, has no missing essential semantic
596 information (ESI), and has no restrictions or references. Type 7, which has the highest complexity and lowest
597 computability, has moderately complex VP, very complex NP and PP/SBAR, has no quantitative information, and
598 has both restrictions and references. The distribution of sentence types is nonuniform; and in total, more than 80% of
599 the sentences are of moderately high to low computability. The intrinsic evaluation results showed an average
600 silhouette coefficient of 0.742 and a cophenetic coefficient of 0.852 for the first-level clustering. The extrinsic
601 evaluation indicated perfect (100%) alignment between the human- and computer-identified types and 80%
602 agreement in the assignment of sentences to the types. It also indicated that sentence types with high computability
603 showed significantly higher information extraction performance than those types with low computability.

604 In their future work, the authors plan to improve the proposed approach and leverage the insights about sentence
605 types in four directions. First, the feature analysis and selection methods could be improved to enhance the
606 performance of clustering and the comprehensiveness of computability analysis. For example, additional features
607 such coreferences, implicit meanings, and difficult-to-interpret concepts (e.g., “structural integrity” is more difficult
608 to interpret than “height”) could be explored. Second, the analysis could be extended to multi-sentence requirements
609 and to other types of codes or regulatory documents. One sentence could include multiple requirements, while
610 multiple sentences could jointly express one requirement. The proposed computability features and indicators could
611 be adapted to analyze multi-sentence requirements. They could also be adapted to analyze other types of codes or
612 other types of regulatory documents such as contract specifications, which tend to be more complex in terms of
613 length, sentence structures, exceptions, and conjunctive and alternative obligations, etc. Third, the proposed features
614 and metrics, along with the identified sentence types, could be used to analyze the capabilities of existing ACC
615 approaches and tools (e.g., the CORENET ePlanCheck, SMARTcodes, SNACC system), in terms of which types of

requirements can be processed, represented, and checked, and which not. This could provide a better understanding of what tools and methods exists and where the practical and knowledge gaps are. Fourth, and most importantly, the authors will explore how the knowledge of sentence types – and their features, complexities, and computability – can help pave the path toward smart and computable codes and eventually full automation in code checking and analysis. As we embark on this endeavor, understanding the types and characteristics of the text we are dealing with is essential to understanding the problem at hand, breaking it into manageable parts, and tackling each part one by one. Our ultimate goal is to leverage machine learning and other artificial intelligence approaches to reach a level where we can automatically process the entire building code and represent it in a computable manner – hopefully including all types of sentences in the code.

Data Availability Statement

Some or all data, models, or code that support the findings of this study are available from the corresponding author upon reasonable request (building-code sentence data and computational code for the clustering-based computability analysis).

Acknowledgements

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

References

[1] AEC3. (2012). International Code Council. http://aec3.homepage.t-online.de/en/5/5_013_ICC.htm. (September 15, 2020).

[2] Aggarwal, C. C., and Zhai, C. (2012). *Mining Text Data*, Springer Science & Business Media, U.S.

[3] Agresti, A. (2003). *Categorical data analysis*. John Wiley & Sons, Hoboken.

[4] Al Qady, M., and Kandil, A. (2014). "Automatic clustering of construction project documents based on textual similarity." *Autom. Construct.*, 42, 36-49.

[5] Alashwal, A.M., and Abdul-Rahman, H. (2014). “Using PLS-PM to model the process of inter-project learning in construction projects.” *Autom. Construct.*, 44, 176-182.

[6] Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E.D., Gutierrez, J.B., and Kochut, K. (2017). “A brief survey of text mining: Classification, clustering and extraction techniques.” arXiv preprint arXiv:1707.02919.

[7] Ambati, B.R., Reddy, S., and Steedman, M. (2016). “Assessing relative sentence complexity using an incremental CCG parser.” *Proc. 2016 Conf. of the North American Chapter of the Association for Computational Linguistics (NAACL): Human Language Technologies*, ACL, Stroudsburg, 1051-1057.

[8] Arthur, D., and Vassilvitskii, S. (2006). *k-means++: The advantages of careful seeding*, Stanford.

[9] Bekkerman, R., El-Yaniv, R., Tishby, N., and Winter, Y. (2001). "On feature distributional clustering for text categorization." *Proc., 24th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Association for Computing Machinery (ACM), New York, 146-153.

[10] Bird, S., Klein, E., and Loper, E. (2009). "Natural language processing with Python: analyzing text with the natural language toolkit." O'Reilly Media, Inc.

[11] Cadez, I., Heckerman, D., Meek, C., Smyth, P., and White, S. (2003). "Model-based clustering and visualization of navigation patterns on a web site." *Data Min. Knowl. Discov.*, 7(4), 399-424.

[12] Clark, V., and Creswell, J. (2008). *The mixed methods readers*, Sage Publications, Thousand Oaks.

[13] Cutting, D. R., Karger, D. R., Pedersen, J. O., and Tukey, J. W. (2017). "Scatter/gather: A cluster-based approach to browsing large document collections." *ACM SIGIR Forum*, 51(2), 148-159.

[14] Dimyadi, J., and Amor, R. (2013). "Regulatory knowledge representation for automated compliance audit of BIM-based models." *Proc., 30th CIB W78 Int. Conf.*, Conseil International du Bâtiment (CIB), Rotterdam, Netherlands, 68-78.

[15] Dimyadi, J., Clifton, C., Spearpoint, M., and Amor, R. (2016). "Computerizing regulatory knowledge for building engineering design." *J. Comput. Civil. Eng.*, 30(5), 10.1061/(ASCE)CP.1943-5487.0000572, C4016001.

[16] Dougherty, J., Kohavi, R. and Sahami, M. (1995). "Supervised and unsupervised discretization of continuous features." *Mach. Learn Proc. 1995*, Morgan Kaufmann, Burlington, 194-202.

[17] Eastman, C., Lee, J., Jeong, Y., and Lee, J. (2009). "Automatic rule-based checking of building designs." *Autom. Construct.*, 18(8), 1011-1033.

[18] Etikan, I., Musa, S.A., and Alkassim, R.S. (2016). "Comparison of convenience sampling and purposive sampling." *Am. J. Theor. Appl. Stat.*, 5(1), 1-4.

[19] El-Diraby, T.E., and Osman, H. (2011). "A domain ontology for construction concepts in urban infrastructure products." *Autom. Construct.*, 20(8), 1120-1132.

[20] Fodeh, S., Punch, B. and Tan, P. N. (2011). "On ontology-driven document clustering using core semantic features." *Knowl. Inform. Syst.*, 28(2), 395-421.

[21] Garrett, J.H., and Fenves, S.J. (1987). "A knowledge-based standards processor for structural component design." *Engineering with Computers*, 2(4), 219-238.

[22] Garrett Jr, J.H. and Hakim, M.M. (1992). "Object-oriented model of engineering design standards." *J. Comput. Civil. Eng.*, 6(3), 323-347.

[23] Hallgren, K. A. (2012). "Computing inter-rater reliability for observational data: an overview and tutorial." *Tutorials in Quant. Methods for Psychol.*, 8(1), 23.

[24] Hjelseth, E., and Nisbet, N. (2010). "Exploring semantic based model checking." *Proc. 27th CIB W78 Int. Conf.* <http://itc.scix.net/data/works/att/w78-2010-54.pdf> (May 15, 2018).

[25] Jin, X.H. (2010). "Neurofuzzy decision support system for efficient risk allocation in public-private partnership infrastructure projects." *J. Comput. Civ. Eng.*, 24(6), 525-538.

[26] Jurafsky, D., and Martin, J. (2014). Speech and language processing. 3rd edn. Pearson, London.

[27] Kaufman, L., and Rousseeuw, P.J. (2009). *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, Hoboken.

[28] Ketchen, D.J., and Shook, C.L. (1996). "The application of cluster analysis in strategic management research: an analysis and critique." *Strateg. Manage. J.*, 17(6), 441-458.

[29] Kifokeris, D., and Xenidis, Y. (2017). "Constructability: Outline of past, present, and future research." *J. Construct. Eng. Manag.*, 143(8), 04017035.

[30] Kilicoglu, H., and S. Bergler. (2009). "Syntactic dependency based heuristics for biological event extraction." *Proc., 2009 Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*, Boulder, Colorado, U.S., 119-127.

[31] Li, S., Cai, H., and Kamat, V.R. (2016). "Integrating natural language processing and spatial reasoning for utility compliance checking." *J. Constr. Eng. Manage.*, 142(12), p.04016074.

[32] Li, Y., Chung, S. M., and Holt, J. D. (2008). "Text document clustering based on frequent word meaning sequences." *Data Knowl. Eng.*, 64(1), 381-404.

[33] Lomakina, L. S., Rodionov, V. B., and Surkova, A. S. (2014). "Hierarchical clustering of text documents." *Autom. Rem. Contr.*, 75(7), 1309-1315.

[34] Malsane, S., Matthews, J., Lockley, S., Love, P. E., and Greenwood, D. (2015). "Development of an object model for automated compliance checking." *Autom. Construct.*, 49, 51-58.

[35] Manning, C.D., Schütze, H., and Raghavan, P. (2008). *Introduction to information retrieval*. Cambridge university press, Cambridge, United Kingdom.

[36] Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J.R., Bethard, S., and McClosky, D. (2014). "The Stanford CoreNLP natural language processing toolkit." *Proc. ACL 2014: system demonstrations*, ACL, Stroudsburg, 55-60.

[37] Massung, S., Zhai, C., and Hockenmaier, J. (2013) "Structural parse tree features for text representation." *2013 IEEE 7th International Conference on Semantic Computing*, IEEE, New York, 9-16.

[38] Naughton, M., Stokes, N., and Carthy, J. (2010). "Sentence-level event classification in unstructured texts." *Information retrieval*, 13(2), 132-156.

[39] Nawari, N.O. (2019). "A Generalized Adaptive Framework (GAF) for Automating Code Compliance Checking." *Buildings*, 9(4), 86.

[40] Ng, H. S., Toukourou, A. and Soibelman, L. (2006). "Knowledge discovery in a facility condition assessment database using text clustering." *J. Infrastruct. Syst.*, 12(1), 50-59.

[41] Ozkaya, I., and Akin, Ö. (2006). "Requirement-driven design: assistance for information traceability in design computing." *Design Studies*, 27(3), 381-398.

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

[43] Rousseeuw, P. J. (1987). "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis." *J. Comput. Appl. Math.,* 20, 53-65.

[44] Salama, D.A., and El-Gohary, N.M. (2013). "Automated compliance checking of construction operation plans using a deontology for the construction domain." *J. Comput. Civ. Eng.*, 27(6), 681-698.

[45] Salton, G., and Buckley, C. (1988). "Term-weighting approaches in automatic text retrieval." *Inf. Process. Mgmt.*, 24(5), 513-523.

[46] Särndal, C. E., Swensson, B., and Wretman, J. (2003). *Model Assisted Survey Sampling*. Springer Science & Business Media, New York, 100-109.

[47] Sarstedt, M., and Mooi, E. (2014). *A Concise Guide to Market Research*. Springer, Berlin.

[48] Shepitsen, A., Gemmell, J., Mobasher, B., and Burke, R. (2008). "Personalized recommendation in social tagging systems using hierarchical clustering." *Proc., 2008 ACM conf. on Recommender Systems*, Association for Computing Machinery (ACM), New York, 259-266.

[49] Sneath, P.H., and Sokal, R.R. (1973). *Numerical taxonomy. The principles and practice of numerical classification*. W.H. Freeman and Company, San Francisco.

[50] Štajner, S., and Hulpuș, I. (2018). "Automatic assessment of conceptual text complexity using knowledge graphs." *Proc. 27th International Conference on Computational Linguistics*, ACL, Stroudsburg, 318-330.

[51] Stemler, S.E. (2004). "A comparison of consensus, consistency, and measurement approaches to estimating interrater reliability." *Pract. Assess. Res. Eval.*, 9(1), 4.

[52] Solibri. (2018). "Solibri Model Checker." https://www.solibri.com/products/solibri-model-checker. (May 15, 2018)

[53] Solihin, W., and Eastman, C. (2015). "Classification of rules for automated BIM rule checking development." *Autom. Construct.*, 53, 69-82.

[54] Valdes-Vasquez, R., and Klotz, L. E. (2012). "Social sustainability considerations during planning and design: framework of processes for construction projects." *J. Construct. Eng. Manag.*, 139(1), 80-89.

[55] Weise, M., Liebich, T., Nisbet, N. and Benghi, C. (2017). "IFC model checking based on mvdXML 1.1." *eWork and eBusiness in Architecture, Engineering and Construction: ECPPM 2016*, 19-26.

[56] Yang, L., Cai, X., Zhang, Y., and Shi, P. (2014). "Enhancing sentence-level clustering with ranking-based clustering framework for theme-based summarization." *Inform. Sci.*, 260, 37-50.

[57] Yao, J., Mao, Q., Goodison, S., Mai, V., and Sun, Y. (2015). "Feature selection for unsupervised learning through local learning." *Pattern Recognit. Lett*., 53, 100-107.

[58] Yurchyshyna, A., and Zarli, A. (2009). "An ontology-based approach for formalisation and semantic organisation of conformance requirements in construction." *Autom. Construct.*, 18(8), 1084-1098.

[59] Zhai, C., and Massung, S. (2016). Text data management and analysis: a practical introduction to information retrieval and text mining, ACM, New York, USA.

[60] Zhang, J., and El-Gohary, N. (2013). "Semantic NLP-based information extraction from construction regulatory documents for automated compliance checking." *J. Comput. Civ. Eng.*, 10.1061/(ASCE)CP.1943-5487.0000346, 04015014.

[61] Zhang, J., and El-Gohary, N. (2015). "Automated information transformation for automated regulatory compliance checking in construction." *J. Comput. Civ. Eng.*, 10.1061/(ASCE)CP.1943-5487.0000427, B4015001.

[62] Zhang, J., and El-Gohary, N. (2016). "Extending building information models semiautomatically using semantic natural language processing techniques." *J. Comput. Civ. Eng.*, 10.1061/(ASCE)CP.1943-5487.0000536, C4016004.

[63] Zhang, J., and El-Gohary, N. (2017a). "Semantic-based logic representation and reasoning for automated regulatory compliance checking." *J. Comput. Civ. Eng.*, 31(1), 10.1061/(ASCE)CP.1943-5487.0000583.

[64] Zhang, J., and El-Gohary, N. (2017b). "Integrating semantic NLP and logic reasoning into a unified system for fully-automated code checking." *Autom. Construct.*, 73, 45-57.

[65] Zhang, R., and El-Gohary, N. (2020). "A machine-learning approach for building-code sentence generation for automatic semantic analysis." *Proc. 2020 ASCE CRC*, ASCE.

[66] Zhong, B., Xing, X., Luo, H., Zhou, Q., Li, H., Rose, T., and Fang, W. (2020). "Deep learning-based extraction of construction procedural constraints from construction regulations." *Adv. Eng. Inform.*, 43, p.101003.

[67] Zhou, P., and El-Gohary, N. (2017). "Ontology-based automated information extraction from building energy conservation codes." *Autom. Construct.*, 74, 103-117.

List of Figure Captions

Fig. 1. Constituency parsing tree of an example sentence.

Fig. 2. Methodology for building-code computability analysis and typing.

Fig. 3. Extrinsic evaluation process.

Fig. 4. Plot of the percentage of variance explained by the clusters.

Fig. 5. Plot of the silhouette coefficient of each sentence and cluster.

Tables

Table 1. Semantic Information Elements for Representing Requirements for Compliance Checking Purposes (Zhang and El-Gohary 2015)

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

Table 2. Scales of Syntactic and Semantic Cluster Complexities and Computability

Symbolic representation	Scale of syntactic cluster complexity, semantic cluster complexity, and syntactic feature complexity	Scale of cluster computability
---	Very simple	Very low
--	Simple	Low
-	Moderately simple	Moderately low
+	Moderately complex	Moderately high
++	Complex	High
+++	Very complex	Very high

Table 3. Cluster Characterization and Computability of First-level Clusters

Cluster	Syntactic features				Syntactic cluster complexity	Semantic features				E	Semantic cluster complexity	Computability
	NP	VP	PP/SBAR	S		A	Q	Rf	Rs			
1	0.11 (--)	0.19 (--)	0.00 (---)	0.12 (--)	0.09 (--)	1.00 (+)	0.98 (+)	0.00 (-)	0.00 (-)	0.00 (-)	0.00 (---)	0.96 (+++)
2	0.09 (---)	0.18 (---)	0.00 (---)	0.10 (---)	0.00 (---)	0.00 (-)	0.00 (-)	0.00 (-)	0.00 (-)	1.00 (+)	0.33 (--)	0.84 (++)
3	0.08 (---)	0.20 (--)	0.00 (---)	0.12 (--)	0.05 (--)	0.00 (-)	0.00 (-)	1.00 (+)	0.00 (-)	1.00 (+)	0.61 (+)	0.67 (+)
4	0.13 (-)	0.22 (-)	0.00 (---)	0.15 (-)	0.19 (-)	1.00 (+)	0.00 (-)	1.00 (+)	0.00 (-)	1.00 (+)	0.61 (+)	0.60 (+)
5	0.15 (+)	0.36 (++)	0.07 (++)	0.31 (++)	0.82 (++)	1.00 (+)	1.00 (+)	0.00 (-)	1.00 (+)	0.00 (-)	0.39 (-)	0.40 (-)
6	0.13 (-)	0.30 (+)	0.05 (--)	0.24 (+)	0.56 (+)	0.00 (-)	0.00 (-)	0.00 (-)	1.00 (+)	1.00 (+)	0.72 (++)	0.36 (-)
7	0.15 (+)	0.31 (+)	0.05 (--)	0.26 (+)	0.64 (+)	0.00 (-)	0.00 (-)	1.00 (+)	1.00 (+)	1.00 (+)	1.00 (+++)	0.18 (--)
8	0.17 (++)	0.39 (+++)	0.08 (+++)	0.35 (+++)	1.00 (+++)	1.00 (+)	1.00 (+)	1.00 (+)	0.96 (+)	0.00 (-)	0.65 (+)	0.17 (--)
9	0.19 (+++)	0.32 (+)	0.08 (+++)	0.29 (++)	0.79 (++)	1.00 (+)	0.00 (-)	1.00 (+)	1.00 (+)	1.00 (+)	1.00 (+++)	0.10 (---)

Note: 1. NP = noun phrase; VP = verb phrase; PP = preposition phrase; SBAR = clause introduced by a subordinating conjunction; S = whole sentence; A = compliance checking attribute; Q = quantitative semantic information; Rf = reference; Rs = restriction; E = percentage of sentences in the cluster that have missing essential semantic information.
2. The scales of the syntactic feature complexity, syntactic and semantic cluster complexities, and computability are shown in Table 2.
3. For the semantic feature complexity: + symbol = feature is present in most of the sentences in the cluster; - symbol = feature is absent from most of the sentences in the cluster.

Table 4. Cluster Characterization and Computability of Children Clusters

Parent cluster	Subclustering features	Comparison of children clusters	Computability of children clusters
Cluster 1	IN CC ('IN', 'DT') Total number of phrase structure rules	Compared to sentences in Cluster 1.1, Cluster 1.2 sentences have more prepositions (IN) (e.g., "of", "in", "at") and/or more conjunctions (CC), and have more phrase structure rules. Sentences in Cluster 1.2 are likely to have multiple requirements (e.g., two requirements, one on width and one on height) in the form of preposition phrases and/or conjunctions and thus are more complex both syntactically and semantically.	Cluster 1.1 (+++) Cluster 1.2 (++)
Cluster 5	NP → NP PP ('DT' 'NN') ('JJ', 'NN') ('DT', 'JJ')	Compared to sentences in Cluster 5.1, Cluster 5.2 sentences have more noun phrases (NP), more pairs of determiner (DT) and noun (NN), and more pairs of adjective (JJ) and noun. Sentences in Cluster 5.2 are likely to have multiple requirements in the form of long and nested noun phrases and thus are more complex both syntactically and semantically.	Cluster 5.1 (+) Cluster 5.2 (-)
Cluster 8	NP → NP PP NP → NNP CD ('CD' 'NNS') ('NNS' 'IN') ('IN' 'NN') ('NN' 'IN') ('JJ' 'NN') ('IN' 'NNP') ('DT' 'NN') ('JJR' 'IN') ('IN' 'DT') ('DT' 'JJ') ('MD' 'VB') ('CD' '.')	Compared to sentences in Cluster 8.1, Cluster 8.2 sentences have more pairs of determiner and noun, pairs of preposition and noun, and pairs of cardinal number (CD) and noun. Sentences in Cluster 8.2 are likely to have multiple requirements in the form of long and nested noun phrases and thus are more complex both syntactically and semantically.	Cluster 8.1 (--) Cluster 8.2 (---)

Note: NNP = proper noun, singular; NNS = noun, plural; JJR = adjective, comparative; MD = modal; VB = verb, base form.

798 **Table 5.** Sentence Types in the International Building Code

Sentence type	Example sentence	Percentage of sentence types in the whole dataset	Computability
1.1 (Cluster 1.1)	The height of door openings shall not be less than 80 inches (2032 mm).	1.5%	Very high (0.96)
1.2 (Cluster 1.2)	Occupiable spaces, habitable spaces and corridors shall have a ceiling height of not less than 7 feet 6 inches.	1.3%	High (0.96)
2 (Cluster 2)	Louvers shall be prohibited.	6.9%	High (0.84)
3.1 (Cluster 3)	Shaft enclosures shall meet the requirements of Section 703.2.1.	7.8%	Moderately high (0.67)
3.2 (Cluster 4)	The fire-resistance rating of building elements, components or assemblies shall be determined in accordance with the test procedures set forth in ASTM E 119 or UL 263 or in accordance with Section 703.3.	3.0%	Moderately high (0.60)
4.1 (Cluster 5.1)	Spacing of braced wall lines shall not exceed 35 feet on center in both the longitudinal and transverse directions in each story.	12.1%	Moderately high (0.40)
4.2 (Cluster 5.2)	Where an egress court serving a building or portion thereof is less than 10 feet in width, the egress court walls shall have not less than 1-hour fire-resistance-rated construction for a distance of 10 feet above the floor of the court.	11.2%	Moderately low (0.40)
5.1 (Cluster 6)	Openings between the Group S-2 enclosed parking garage and Group S-2 open parking garage, except exit openings, shall not be required to be protected.	27.7%	Moderately low (0.36)
5.2 (Cluster 7)	Where exterior walls serve as a part of a required fire-resistance-rated shaft or exit enclosure, or separation, such walls shall comply with the requirements of Section 705 for exterior walls and the fire-resistance-rated enclosure or separation requirements shall not apply.	19.2%	Low (0.18)
6.1 (Cluster 8.1)	In occupancies in Groups B, E, F, I-1, M, R-l, R-2, R-4, Sand U, where the building is equipped throughout with an automatic sprinkler system in accordance with Section 903.3.1.1, the length of the dead-end corridors shall not exceed 50 feet.	3.8%	Low (0.17)
6.2 (Cluster 8.2)	The aggregate floor area of the Group H occupancies located at the perimeter of the unlimited area building shall not exceed 10 percent of the area of the building nor the area limitations for the Group H occupancies as specified in Table 503 as modified by Section 506.2, based upon the percentage of the perimeter of each Group H floor area that fronts on a street or other unoccupied space.	4.0%	Very low (0.17)
7 (Cluster 9)	In determining the fire-resistance rating of exterior bearing walls, compliance with the ASTM E 119 or UL 263 criteria for unexposed surface temperature rise and ignition of cotton waste due to passage of flame or gases is required only for a period of time corresponding to the required fire-resistance rating of an exterior nonbearing wall with the same fire separation distance, and in a building of the same group.	1.5%	Very low (0.10)

799

800

Table 6. Average Silhouette Coefficients for the Tested Clustering Algorithms

Clustering		Hierarchical clustering							K-means	PAM
		Single	Complete	Average	McQuitty	Median	Centroid	Ward's		
First-level clustering		0.464	0.647	0.678	0.701	0.129	0.478	**0.742**	0.488	0.485
Subclustering	Cluster 1	0.527	**0.674**	0.545	**0.674**	0.539	0.545	**0.674**	0.601	**0.674**
	Cluster 5	0.566	**0.661**	0.568	0.589	0.589	0.568	0.613	0.642	0.643
	Cluster 8	0.234	0.468	0.463	**0.515**	0.469	0.463	0.483	0.498	0.506

Note: Bold font indicates the best performance(s); PAM = partition around medians.

Table 7. Average Silhouette Coefficients Before and After Feature Selection and Synthesis

Clustering		Initial features	Final features
First-level clustering		0.183	0.742
Subclustering	Cluster 1	0.245	0.674
	Cluster 5	0.245	0.661
	Cluster 8	0.233	0.515

Table 8. Experimental Results of Application-Oriented Evaluation

	Sentences with very high or high computability			Sentence types with low or very low computability		
Sentence types	Type 1.1	Type 1.2	Type 2	Type 6.1	Type 6.2	Type 7
Precision	0.95	0.96	0.98	0.81	0.79	0.77
Recall	0.98	0.95	0.97	0.79	0.78	0.75
F1-measure	0.96	0.96	0.97	0.80	0.78	0.76