Domain-Specific Hierarchical Text Classification for Supporting Automated Environmental Compliance Checking

Peng Zhou, S.M.ASCE[1]; and Nora El-Gohary, A.M.ASCE[2]

Abstract: Automated environmental compliance checking requires automated extraction of rules from environmental regulatory textual documents such as energy conservation codes and EPA regulations. Automated rule extraction requires complex text processing and analysis for information extraction and subsequent formalization of the extracted information into computer-processable rules. In the proposed automated compliance checking (ACC) approach, the text is first classified into predefined categories before information extraction (IE). The advantages are that irrelevant text will be filtered out during text classification (TC) and text with similar semantic meaning will be grouped, thereby improving the efficiency and accuracy of further IE and compliance reasoning (CR). The categories used for TC are predefined in a semantic TC topic hierarchy, and the classified text is subsequently used in semantic IE and semantic CR. This paper presents the proposed machine learning (ML)-based TC algorithm for classifying clauses in environmental regulatory documents based on the TC topic hierarchy. In developing the algorithm, different text preprocessing techniques, ML algorithms, and performance improvement strategies were tested and used. The final TC algorithm was tested on 10 environmental regulatory documents and evaluated in terms of precision and recall. The algorithm achieved approximately 97 and 84% average recall and precision, respectively, on the testing data. **DOI: 10.1061/(ASCE)CP.1943-5487.0000513.** *© 2015 American Society of Civil Engineers.*

Author keywords: Automated compliance checking; Semantic systems; Automated construction management systems; Natural language processing; Text classification; Machine learning.

Introduction

Compliance checking aims to ensure the compliance of a project with applicable norms such as laws, regulations, codes, and contractual requirements. Manual compliance checking is a time-consuming and error-prone task (Eastman et al. 2009; Tan et al. 2010). Automated compliance checking (ACC) has, therefore, attracted much research effort to reduce the cost and time of this task. Examples of the most recent efforts in the area of ACC in construction include (1) using an ontology-based approach to model regulatory constraints for supporting construction quality compliance checking (Zhong et al. 2012); and (2) using manually encoded rules for building design checking (Eastman et al. 2009; Tan et al. 2010; Nawari 2012). Despite the importance of these efforts, they require manual extraction of requirements from textual regulatory documents (e.g., codes) and encoding of these requirements in a computer-processable rule format, thereby making the compliance checking process not entirely automated.

To address this gap, in their previous work, the authors proposed a hybrid deontic-based and natural language processing (NLP)-enabled approach for automated regulatory and contractual compliance checking in construction. A deontic model for ACC in construction (a semantic model based on the theory of rights and obligations) was proposed to support normative automated reasoning (Salama and El-Gohary 2013a). The NLP techniques, including text classification (TC) (Salama and El-Gohary 2013b; Zhou and El-Gohary 2014) and information extraction (IE) (Zhang and El-Gohary 2013), were proposed to support automated text processing and analysis for automated IE from regulatory and contractual documents and subsequent formalization of the extracted information into computer-processable rules. In the proposed ACC approach, the text is first classified into predefined categories before IE. The advantages are that irrelevant text will be filtered out during TC and text with similar semantic meaning will be grouped, thereby improving the efficiency and accuracy of further IE.

In this research, the authors build on their previous work in NLP (Salama and El-Gohary 2013b) and TC in seven main ways. First, the classification and text features of a different type of document are explored—environmental regulatory documents instead of general conditions of contractual documents. Second, and more importantly, a deeper TC granularity level is addressed. This research deals with a more detailed level of TC topics–classifying the text according to topics in the fifth level of the hierarchy instead of the first level. As you go to a more specialized level of topics, TC becomes typically more challenging, because the levels of knowledge and terminology of the text become more specialized and more specific, thereby making the text harder to discriminate (Silla and Freitas 2011). Third, a multiclass classification model is used, to deal with the multilabel classification problem, instead of a simple binary classification model. Fourth, the use of domain-specific stopword lists is explored as a means for domain adaptation. Fifth, two text representation methods are tested and evaluated: a commonly used text representation method [bag of words (BOW) model] and a more complex method that can capture partial text semantics (bigram model). Sixth, two supervised term weighting

[1]Graduate Student, Dept. of Civil and Environmental Engineering, Univ. of Illinois at Urbana-Champaign, 205 N. Mathews Ave., Urbana, IL 61801.

[2]Assistant Professor, Dept. of Civil and Environmental Engineering, Univ. of Illinois at Urbana-Champaign, 205 N. Mathews Ave., Urbana, IL 61801 (corresponding author). E-mail: gohary@illinois.edu

Note. This manuscript was submitted on September 8, 2014; approved on May 1, 2015; published online on September 24, 2015. Discussion period open until February 24, 2016; separate discussions must be submitted for individual papers. This paper is part of the ***Journal of Computing in Civil Engineering***, © ASCE, ISSN 0887-3801/04015057(14)/$25.00.

schemes [term frequency relevance frequency (TFRF) and term frequency maximum relevance frequency ($TF_{max}RF$)] that are used for binary classification are modified (to $TFRF_M$ and $TF_{max}RF_M$, where the subscript M means modified) to adapt them to multiclass classification problems, and are tested and evaluated in comparison with the commonly used unsupervised term weighting scheme [term frequency inverse document frequency (TFIDF)]. Seventh, more ML algorithms are tested, and some important parameters are tuned for each algorithm, in an effort to improve the process of selecting the best ML algorithm (including its parameters).

The following sections of the paper present the proposed domain-specific, ML-based hierarchical TC algorithm for classifying environmental clauses according to a set of predefined, semantic labels. The labels are defined based on a hierarchy of topics in the ACC deontic model. The problem is formulated as a multilabel classification problem because one clause could be assigned more than one label. In developing the algorithm, the performance of a number of popular ML algorithms and NLP techniques (data preprocessing, feature selection, and stopword removal) were tested and evaluated.

Background

Text Classification Problems

NLP is a subfield of artificial intelligence that aims to enable computers to process natural language in a human-similar way (Manning and Schutze 1999). TC is a subfield of NLP that aims to assign documents (or text units such as paragraphs or clauses) to one or more predefined categories (Manning and Schutze 1999). The text is usually unstructured (i.e., does not have a clear computer-readable structure). A category is represented by a label and may refer to a class or concept. TC problems can be categorized as multilabel or single-label classification problems (Tsoumakas and Katakis 2007; Ghamrawi and McCallum 2005). Multilabel classification aims to assign more than one label to a document. In contrast, single-label classification aims to predict only one label for each document. A single-label classification problem can be further categorized as (1) a binary classification problem, if there are only two classes (usually as a positive class and a negative class) in the data set; or (2) a multiclass classification problem, if the number of classes is more than two.

There are two common methods to solve multilabel classification problems (Tsoumakas and Katakis 2007). The less commonly used method is the algorithm adaptation method (AAM), which can cope with multilabel classification problems directly by modifying or extending some available algorithms. The advantage of AAM is that it can predict a set of labels at one time. However, its performance is still not good enough (Tsoumakas and Katakis 2007). The most commonly used method is the problem transformation method (PTM), in which a multilabel classification problem can be transformed into two or more single-label classification problems by assuming the independence of labels. If the number of labels in the original data sets is L, the transformation will result in L single-label classification problems (and thus L classifiers) and L number of data sets (one data set for each label L_i). Each data set is used to train one classifier on predicting the label L_i of that data set. During testing, each test clause is processed by those L number of classifiers one by one, in which each classifier decides whether to assign its corresponding label L_i or not. The total number of assigned labels during this process form the final label set of this test clause.

A multilabel classification problem that was transformed to a single-label problem can be further addressed using a binary classification or a multiclass classification approach (Aly 2005). For each of the transformed data sets with label L_i, a binary classification approach defines the label L_i as the positive class and combines all other labels in a negative class, and then applies binary classification algorithms to address this binary classification problem. In contrast, a multiclass classification approach does not combine the labels and directly uses multiclass classification algorithms. The advantage of a multiclass classification approach is that data imbalance problems resulting from the transformation (i.e., combining labels results in a relatively larger negative data set in comparison to the smaller positive data set) could be avoided.

Text Classification Using Machine Learning Techniques

ML techniques are commonly used for TC. ML refers to a system learning from available data or previous experience (Manning and Schutze 1999). ML techniques can be categorized into three main types: (1) supervised ML: human guidance is provided in the form of labeled documents (all documents are given one or more predefined labels), in which a training data set is used to train the classifier to automatically classify a given document according to a predefined set of labels, and a testing data set is used to test the performance of the classifier; (2) unsupervised ML: documents are not labeled for training; and thus, instead of classifying given documents according to a predefined set of labels, classifiers automatically (and without human guidance) cluster documents into potentially useful categories; and (3) semisupervised ML: only a fraction of the training data set is labeled, which provides partial human guidance. In comparison to unsupervised and semisupervised ML, supervised ML algorithms require higher manual effort for preparing the training data set. However, their precision and recall are typically higher because of the benefit from human guidance. In this paper, a supervised ML-based approach is adopted because of the desired high recall results.

ML TC requires the representation of documents in terms of numerical features. The most commonly used method for representing features of the text is the BOW model (Manning and Schutze 1999). In this model, a document is represented as an unordered set of words along with their corresponding frequencies of occurrence in this document, and the positions of the words are ignored. The words are all drawn from the vocabulary used in the document. The frequency of each word is then normalized by the total occurrence of this word in the whole document collection. The advantages of the BOW model are simplicity and computational efficiency, although they come at the cost of discarding the relationships among words in terms of their relative positions in the document. Another, but less commonly used, text representation method is the bigram model (Manning and Schutze 1999). In this model, the semantic relationships between any two adjacent words are captured. For example, the word group "spring thermal radiation" is more likely to occur than "thermal spring radiation" for the building energy efficiency topic. A document is represented by all such adjacent pairs of words along with their corresponding frequencies of occurrence in this document. A word-pair frequency in one document is then normalized by its total frequency in the entire set of documents.

Because different features have different powers in indicating a category, they should be assigned different weights. There are two types of weighting schemes: unsupervised term weighting and supervised term weighting. Membership information refers to the known information about which category a training document belongs to. Unsupervised term weighting does not use this information. The most state-of-the-art unsupervised weighting scheme is TFIDF (Manning et al. 2009). Term frequency (TF) refers to the total occurrence frequency of a term in one document; document frequency (DF) refers to the number of documents in the entire document collection that contains this term; and IDF refers to the inverse of DF. TFIDF assumes that (1) if a term occurs frequently in one document, then it is highly relevant to the category of this document, and (2) if a term occurs frequently in many documents in the collection, then it is probably not discriminative of any category of documents. Accordingly, TFIDF aims to assign (1) a higher weight to a term that appears frequently in one document, and (2) a lower weight to a term that appears frequently in many documents in a collection.

In contrast to unsupervised term weighting, membership information is used in supervised term weighting. Because not all categories have the same number of documents, supervised term weighting takes this statistical document distribution information into account when calculating the weight of a term in a document. Examples of newly developed supervised term weighting schemes include TFRF (Man et al. 2009) and logarithmic $\mathrm{TF}_{\max}\mathrm{RF}$ (Xuan and Quang 2014), in which TF is same as that in TFIDF weighting and RF measures the relevance of a term to a category.

Because not all features contribute to the discrimination of a category, nondiscriminative features need to be filtered out to enhance the power of those discriminative features. Feature selection (Manning et al. 2009) is the process of selecting a subset of the features in the training data set and using this subset of features to represent the text. There are two main advantages of implementing feature selection. First, the computational efficiency can be improved by selecting a fraction of the features, especially in cases in which the feature size can be in the order of millions and/or when using algorithms that require expensive computation like Naïve Bayes (NB) algorithms. Second, as mentioned previously, performance can be improved by reducing nondiscriminative features and keeping the most discriminative features. There are two main approaches to selecting features: univariate feature selection (UFS) and recursive feature selection (RFS).

UFS tries to use univariate statistical tests to select features. UFS involves calculating a score for each feature using a scoring function, ranking features based on the scores, and then selecting the best features based on the ranking. To evaluate whether a feature is helpful in representing a category, a utility function is defined as U (feature, category) for scoring features. Feature scoring is the process of ranking features based on a utility function U (feature, category). All features ranked below a predefined threshold are discarded and only the features above the ranking threshold are used in classification. Common feature scoring functions used for multiclass classification include Chi-square (CHI), information gain (IG), and mutual information (MI). For the details of these feature scoring methods, the readers are referred to Aggarwal and Zhai (2012).

Instead of using a scoring function to rank and select features, RFS applies a ML algorithm to select features based on the ranking of features in terms of weights. The ML algorithm is used to assign weights to the features for ranking. The initial feature set is used as training data for the ML algorithm. The learned classifier assigns a weight to every feature. Then, a predefined number of features (N) with the lowest absolute weights are discarded. The remaining features are used as new training data for the ML algorithm. Then the weight of each feature is updated by applying the ML algorithm again on the new training data and another N features with the lowest weights are pruned. This recursive process terminates when the total number of remaining features reaches another predefined number (M).

Hierarchical Text Classification

TC could be flat (nonhierarchical) or hierarchical. Different from flat TC, in hierarchical TC, the labels are organized into a class hierarchy (usually represented as a tree structure or class taxonomy) (Silla and Freitas 2011; Fagni and Sebastiani 2010; Yoon et al. 2006; Sun et al. 2003; Sun and Lim 2001). Representing the labels in the form of a class hierarchy may support the TC process by offering a better description of the meanings of the labels in terms of its superclasses and subclasses. Hierarchical TC problems can be addressed using one of the following three approaches: flat approach, local classifier approach, and global classifier approach (Silla and Freitas 2011). The flat approach does not take the hierarchical information into account and only uses the labels of the leaf classes (Silla and Freitas 2011). When a leaf class is assigned to a document, all of its superclasses are also assigned to that document. This provides a simple but indirect solution to hierarchical TC.

The local classifier approach takes local hierarchical information into account (Silla and Freitas 2011; Yoon et al. 2006; Sun et al. 2003; Sun and Lim 2001). It takes a top-down approach in assigning documents to classes; for each document, the classifier assigns its first level class, then proceeds to assign the document to the direct subclasses of that class, and so on, until reaching the leaf level of the hierarchy. The main disadvantage of the local classifier approach is that a misclassification of one class would propagate down the hierarchy to all subclasses. This may lead to low performance results at the lower levels of the hierarchy.

The global classifier approach takes the class hierarchy as a whole into account (Silla and Freitas 2011; Yoon et al. 2006; Sun et al. 2003). It tries to build a single classifier that can predict all classes from all levels of the hierarchy at one time. Global classifiers are relatively complex and their performances are usually inconsistent. This has limited the application of global classifiers.

In this research a flat approach is used; all labels used for classification are leaf classes. Although flat TC is used, retrieving documents on a parent level is easily achieved by aggregating the retrieved documents that have been retrieved on the leaf/children levels.

State of the Art and Knowledge Gaps in Text Classification

A variety of ML-based TC algorithms (e.g., Aggarwal and Zhai 2012) have been developed in the computer science (CS) domain. Some common methods and popular algorithms implementing these methods include (1) decision trees (DT) method implemented in algorithms of iterative dichotomiser3 (ID3), classifier4.5 (C4.5), classifier5 (C5), and classification and regression trees (CART) (Breiman et al. 1984); (2) probabilistic method implemented in NB algorithm; (3) linear and nonlinear method implemented in support vector machine (SVM) algorithm with linear and radial basis function (rbf) kernel; (4) proximity-based method implemented in algorithms of nearest neighbor and nearest centroid; and (5) ensemble method implemented in algorithms of random forest (RF) and gradient boosted regression trees (GBRT). For the details of these methods, the readers are referred to Aggarwal and Zhai (2012), Breiman (2001), and Friedman (2001). Similar to other TC problems, a number of ML algorithms were explored for multiclass classification problems. For example, Malkani and Gillie (2012) used SVM and NB to classify tweets into a set of topics, Wu et al. (2007) used NB and k-nearest neighbors (kNN) to classify news stories, and Giorgetti and Sebastiani (2003) used NB and SVM to classify answers of open-ended questions in surveys.

Despite of these enormous efforts in the CS domain, many challenges still exist in constructing classifiers that can be effective across different domains and, thus, TC models remain highly domain specific (Blitzer et al. 2007). There is no single best TC algorithm across all domains; the performance of one best performing ML algorithm tested on one data set is not necessarily the best one when tested on another data set, especially when data sets from different domains are more dissimilar (Sebastiani 2002). As discussed in Salama and El-Gohary (2013b), it is difficult to reuse an existing classifier from one domain to another (e.g., medical versus construction), from one subdomain to another (e.g., safety versus environmental), or from one application to another (e.g., document management versus ACC), because text features vary across domains and subdomains, and performance requirements vary across applications (e.g., for ACC, unlike other applications, recall is more critical than precision). There is, thus, a need to identify the specific features of domain text and how to adapt or tune a classifier to those specific features and to the specific performance requirements of the domain or application. A construction-domain-specific TC algorithm is, thus, required for classifying construction documents.

A number of research efforts in the construction domain focused on TC (e.g., Caldas et al. 2002; Kovacevic et al. 2008; Mahfouz 2011; Salama and El-Gohary 2013b). However, hierarchical TC work in the construction domain is limited in the following ways: (1) the performance of hierarchical TC tends to drop quickly when reaching a deeper level in the hierarchy. For example, Caldas and Soibelman (2003) addressed a three-level multilabel binary classification problem, but the accuracy dropped from 96% at the first level to 86% at the third level; (2) the algorithms can only handle single-label classification problems that were transformed from a multilabel problem using a binary classification approach. Dealing with a transformed multilabel classification problem as a binary instead of a multiclass classification problem may encounter data imbalance problems. A data imbalance problem occurs when the documents of one class are much more than the documents of another class(es) (Sun et al. 2007); (3) the algorithms are not sufficiently adapted to the domain. It is important to utilize the features and methods that work best for each domain (Blitzer et al. 2007). For example, domain-specific stopwords could be removed to make domain content-bearing words more discriminative; (4) the types of ML algorithms that were tested and evaluated are limited. For example, to the best of the authors' knowledge, the performance of ensemble methods in classifying construction text were not tested; and (5) the types of term weighting schemes that were tested and evaluated are also limited. Some newly developed supervised term weighting schemes that showed effectiveness in some domains (e.g., Xuan and Quang 2014) were not tested in classifying construction text.

To address these gaps, this research explores the following: (1) the use of multiclass classification approach to deal with multilabel classification problems; (2) the use of a domain-specific stopword list as an approach for domain adaptation; (3) the testing of a number of ML algorithms (e.g., RF and GBRT algorithms that implement the ensemble method) and term weighting schemes that were not commonly evaluated in the construction domain; and (4) the effect of feature selection and domain-specific stopword removal on the performance of hierarchical classification of environmental regulatory documents.

Methodology for Domain-Specific Hierarchical Text Classification of Environmental Regulatory Documents

The TC methodology is summarized in Fig. 1. The TC label hierarchy (Step 1) is part of the deontic ACC model [the presentation of the model is outside of the scope of this paper, and is presented in Salama and El-Gohary (2013a)]. Steps 4 and 5 are iterative. Feature selection and domain-specific stopword removal are tested as potential performance improvement strategies.

Step 1: TC Topic Hierarchy Development

This paper focuses on analyzing the energy efficiency topic, which is a subtopic of the environmental topic (as per Fig. 2). To develop the topic hierarchy, the established methodologies for taxonomy development (e.g., El-Gohary and El-Diraby 2010) were followed. The concepts were extracted based on a review of the main relevant documents in the domain [e.g., environmental codes and standards such as the 2012 International Energy Conservation Code (ICC 2012) and the 2010 ASHRAE Energy Standard for Buildings Except Low-Rise Residential Buildings (ASHRAE 2010)]. Subsequently, the concepts were structured into a taxonomy using a combination of top-down and bottom-up approaches. The commercial building energy efficiency topic subhierarchy is shown in Fig. 2. All the leaf nodes (10 subtopics) were used as labels of classification.

Step 2: Data Preparation and Multilabel Classification Problem Transformation

Approximately 1,200 clauses were collected from 10 regulatory documents (Fig. 3). These documents were selected because they all cover energy efficiency requirements, which is the focus of this paper. In dividing a document into clauses, the document was split to the most granular subheading level. One problem that is generally faced in automatically splitting data is data noise elimination. Usually, original data are in different formats (e.g., .txt, .pdf, and .doc) and/or different encodings (e.g., ANSI and Unicode), whereas a software system can only process files in a certain format. The collected set of clauses were transformed into .txt format as required by the developed TC system. However, noise like unknown characters that occur during transformation to .txt format could undermine the performance of TC. The noise was reduced by automatically transforming different encodings to the UTF-8 encoding.

Data sufficiency is also, generally, another challenge for ML-based TC. There is no set definition of how much data are considered sufficient. In the construction domain, especially, there is no benchmark of what is a sufficient data size. However, generally, the

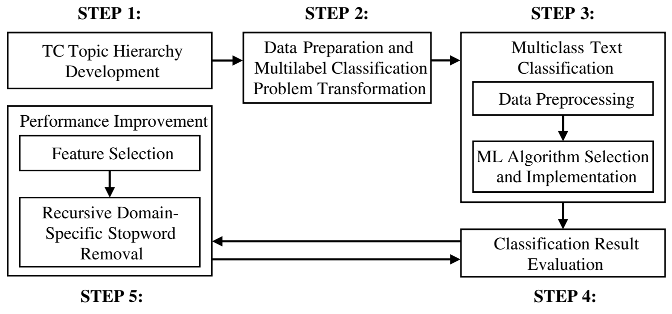
STEP 1:
TC Topic Hierarchy Development
STEP 2:
Data Preparation and Multilabel Classification Problem Transformation
STEP 3:
Multiclass Text Classification
Data Preprocessing
ML Algorithm Selection and Implementation
Performance Improvement
Feature Selection
Recursive Domain-Specific Stopword Removal
Classification Result Evaluation
STEP 5:
STEP 4:

Fig. 1. Methodology for domain-specific hierarchical text classification

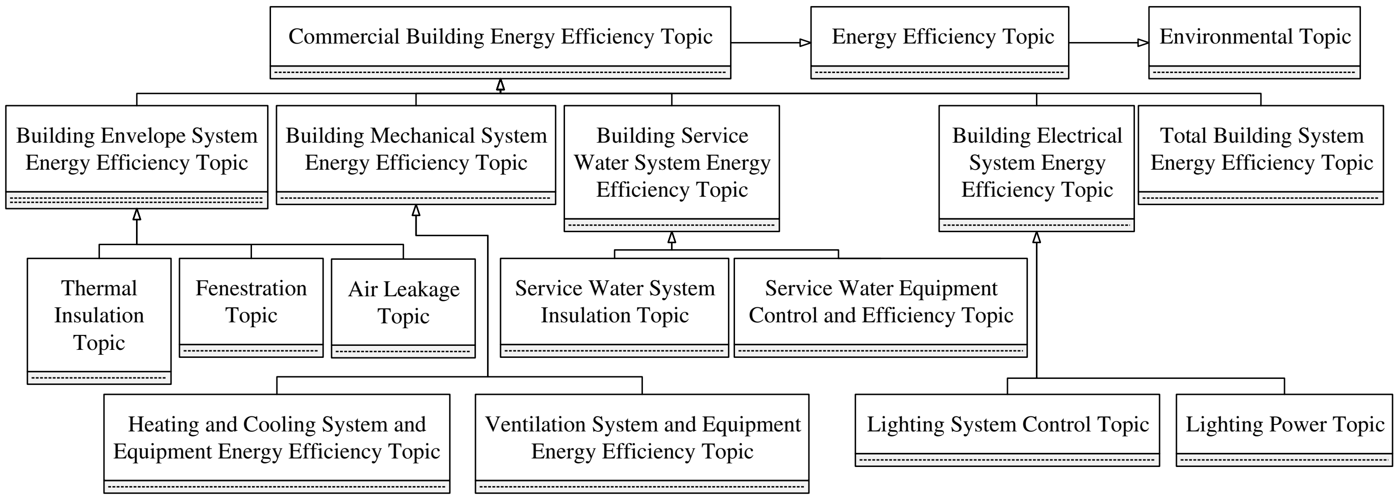
Commercial Building Energy Efficiency Topic
Energy Efficiency Topic
Environmental Topic
Building Envelope System Energy Efficiency Topic
Building Mechanical System Energy Efficiency Topic
Building Service Water System Energy Efficiency Topic
Building Electrical System Energy Efficiency Topic
Total Building System Energy Efficiency Topic
Thermal Insulation Topic
Fenestration Topic
Air Leakage Topic
Service Water System Insulation Topic
Service Water Equipment Control and Efficiency Topic
Heating and Cooling System and Equipment Energy Efficiency Topic
Ventilation System and Equipment Energy Efficiency Topic
Lighting System Control Topic
Lighting Power Topic

Fig. 2. Text classification topic hierarchy

more data collected, the more confident it is believed that the data are sufficient. A series of popular data sets in the CS domain include kdd 2010 and 20 Newsgroups (Chang and Lin 2011). The main properties (number of classes, data size, and number of features) for popular data sets have varied, for example, from 2 to 105 classes, 44 to 10,000 data pieces, and approximately 7,200 to 55,000 text features (Chang and Lin 2011). After data preprocessing (Step 3), the used data set was composed of 10 classes (or topics), approximately 1,200 data pieces (or clauses), and 4,200 text features. Compared with popular data sets in other applications, the number of features in this set is relatively small. However, it is considered sufficient for the following reasons: (1) the vocabulary used in environmental regulations is relatively standardized; and, thus, the number of distinctive features (e.g., "wattage," "daylight," "switch," "insulation," "leakage," and "ventilation") for each class (topic) is relatively small. As a result, a small feature size would result in including sufficient features to identify a text; and (2) the length (number of words) of a clause is relatively small; and, typically, the number of distinctive features of a data piece is proportional to its length. The number of clauses collected in the experiment is also considered sufficient because of the relatively high performance that the classifier achieved.

After data collection, each clause was manually labeled with one or more of the 10 topics, which were identified in Step 1. Which labels should be assigned to a clause is based on analyzing the content of that clause. For example, the following clause was assigned the labels air leakage topic and thermal insulation topic, because it contains requirements for high-pressure ducts in terms of thermal insulation and sealing to prevent air leakage: "Ducts designed to operate at static pressures in excess of 3 in. water gauge (w.g.) (750 Pa) shall be insulated and sealed in accordance with Section C403.2.7" (ICC 2012). For convenience of computer processing, each of the 10 labels was represented by a unique serial number from 1 to 10. The labeling of the data set was reviewed by two other researchers, and 100% agreement on labeling was achieved.

After data labeling, the multilabel classification problem was transformed to 10 (because $L = 10$ in the original data set) single-label multiclass classification problems.

Step 3: Multiclass Text Classification

Data Preprocessing

Data preprocessing is the process of transforming the raw text into the format required by the ML algorithm(s). Most ML algorithms require fixed size numerical feature vectors as the input. Therefore, raw text documents need to be represented by numerical value features. In developing the proposed algorithm, both the BOW model and the bigram model were tested to determine the best method for representing environmental regulatory text.

To represent text using either the BOW model or the bigram model, three commonly used techniques for data preprocessing were implemented: (1) tokenization: tokenization is the task of segmenting the text into pieces called tokens (e.g., words and punctuation), eliminating certain characters such as punctuation, and

Document
2012 International Energy Conservation Code
2010 California Energy Code
ANSI/ASHRAE/IES Standard 90.1-2010 Energy Standard for Buildings Except Low-Rise Residential Buildings
2013 Nonresidential Compliance Manual for the 2013 Building Energy Efficiency Standards
2007 National Green Building Standard, Chapter 7 Energy Efficiency
ANSI/ASHRAE/USGBC/IES Standard 189.1-2009 Standard for the Design of High-Performance Green Buildings Except Low-Rise Residential Buildings
2009 LEED Reference Guide for Green Building Design and Construction
2013 Energy Policy and Conservation Act, Section 342
Energy Independence and Security Act of 2007
2011 North American Fenestration Standard/Specification for Windows, Doors and Skylights

Fig. 3. List of regulatory documents

transforming words to their lowercase forms (e.g., "building, Thermal Insulation" is tokenized into "building thermal insulation"); (2) stopword removal: stopwords refer to those high-frequency and low-content words that are not discriminative in classification such as function words like "am," "is," "a," "the," and "of". According to Zipf's law in NLP (Manning and Schutze 1999), medium and low-frequency words are usually content bearing and thus have higher discriminating power, whereas high-frequency words are low-content bearing and thus have lower discriminating power. Removing stopwords can, thus, help eliminate nondiscriminative high-frequency words, thereby reducing the number of features and revealing the discriminative words; and (3) stemming: stemming is the process of stripping off word suffixes (in some cases prefixes) to map a word to its root or stem. For example, "insulation" and "insulated" can both be mapped to "insul". Stemming reduces the number of features by combining words sharing the same stem. It is usually effective in improving the performance of classification (e.g., Liao et al. 2003). In the proposed TC algorithm, stemming was implemented because the authors' previous experimental work (Salama and El-Gohary 2013b) showed improved performance with stemming. A Python implementation of Porter2 stemming algorithm (Porter 2006) for English stemming was used.

Even if all contentless or low-content features are filtered out, heuristically, not all remaining content-bearing features would have the same power in predicting a label for a clause. Feature weighting is, therefore, used to differentiate between features that are important for classification and those that are not. In developing the proposed algorithm, one unsupervised (TFIDF) and two supervised term weighting schemes (TFRF and $\mathrm{TF}_{\max}\mathrm{RF}$) were tested (Man et al. 2009).

There are many variances of TFIDF weighting schemes. In this research, Eq. (1) was selected because it can prevent overweighting a high TF and DF by using a logarithmic function, where tf_d = frequency of a term in one document or clause d; N = total number of documents or clauses in the collection; and tf_N represents the total frequecy of this term in all documents or clauses

$$\mathrm{TFIDF} = \log(tf_d + 1) \times \left[1 + \log\left(\frac{N}{tf_N}\right)\right] \tag{1}$$

For the two supervised weighting schemes, because both are developed for binary classification, the weighting equations were modified to adapt them to multiclass classification (and were called $\mathrm{TFRF_M}$ and $\mathrm{TF_{max}RF_M}$). In both TFRF and $\mathrm{TF_{max}RF}$, TF measures the term frequency in the same way as in TFIDF, whereas RF and ${}_{\max}\mathrm{RF}$ involve supervised effort in contrast to the unsupervised IDF. In TFRF, relevance frequency (RF) measures how relevant a term is to a category. In binary classification, the RF of a term T in the positive category is the ratio of the number of documents (DF) containing term T in the positive category to that in the negative category [as per Eq. (2)]. Because this research deals with multiclass classification, this original RF was modified (and was called $\mathrm{RF_M}$). The $\mathrm{RF_M}$ of a term T in a category C is the ratio of the number of documents (DF) containing term T in category C to that in all other categories [as per Eq. (3)]. Accordingly, TFRF was modified to $\mathrm{TFRF_M}$, as per Eqs. (4) and (5), respectively. In $\mathrm{TF_{max}RF_M}$ [Eq. (7)], ${}_{\max}\mathrm{RF_M}$ is the maximum $\mathrm{RF_M}$ of a term T in each category C_j [as per Eq. (6)], where the upper bound of j represents the total number of categories. For implementing $\mathrm{TFRF_M}$ and $\mathrm{TF_{max}RF_M}$ [Eqs. (5) and (7), respectively] in multiclass classification, a logarithmic TF function [to prevent overweighting of common, nondiscriminative terms [same as in Eq. (1)] and logarithmic $\mathrm{RF_M}$ and ${}_{\max}\mathrm{RF_M}$ functions [based on the original equations (Man et al. 2009; Xuan and Quang 2014)] were used. Accordingly, Eqs. (8) and (9) were used for implementing these two modified supervised term weighting schemes, where tf_d is the frequency of a term in one document (i.e., clause) d, a is the number of clauses in category C_i containing this term, c is the number of clauses of all other categories containing this term, and the upper bound of i represents the total number of categories.

$$\mathrm{RF} = \frac{\text{DF in positive category containing } T}{\text{DF in negative category containing } T} \tag{2}$$

$$\mathrm{RF_M} = \frac{\text{DF in category } C \text{ containing } T}{\text{DF in all categories except } C \text{ containing } T} \tag{3}$$

$$\mathrm{TFRF} = \mathrm{TF} \times \mathrm{RF} = \mathrm{TF} \times \frac{\text{DF in positive category containing } T}{\text{DF in negative categoriy containing } T} \tag{4}$$

$$\begin{aligned}\mathrm{TFRF_M} &= \mathrm{TF} \times \mathrm{RF_M} \\ &= \mathrm{TF} \times \frac{\text{DF in category } C \text{ containing } T}{\text{DF in all categories except } C \text{ containing } T}\end{aligned} \tag{5}$$

$$\begin{aligned}&{}_{\max}\mathrm{RF_M} \\ &\quad = \text{maximum of set}\left(\frac{\text{DF in category } C_j \text{ containing } T}{\text{DF in all categories except } C_j \text{ containing } T}\right)\end{aligned} \tag{6}$$

$$\begin{aligned}&\mathrm{TF_{max}RF_M} = \mathrm{TF} \times {}_{\max}\mathrm{RF_M} = \mathrm{TF} \\ &\quad \times \text{maximum of set}\left(\frac{\text{DF in category } C_j \text{ containing } T}{\text{DF in all categories except } C_j \text{ containing } T}\right)\end{aligned} \tag{7}$$

$$\mathrm{TFRF}_{M_{C_i}} = \log(tf_d + 1) \times \log\left(10 + \frac{a}{c}\right) \tag{8}$$

$$\mathrm{TF}_{\max_{C_i}}\mathrm{RF}_M = \log(tf_d + 1) \times \max_{C_i}\left[\log\left(10 + \frac{a}{c}\right)\right] \tag{9}$$

A preprocessing program for executing the aforementioned data preprocessing subtasks was coded in Python programming language. The input to the program are raw text files (collected clauses in .txt format), and the output are two data sets (training data set and testing data set) in the Library for Support Vector Machines (LIBSVM) format. Each line in the training and testing data set files represents one clause in feature-numeric value pairs and its corresponding topic serial number (topics are numbered from 1 to 10). Because the number of clauses for each topic varies very differently (from approximately 30 to 180, Fig. 4), the input files (1,215 collected clauses) were randomly split into training and testing data sets, with a ratio of 2:1, respectively, to avoid a very small testing data set size. Because one classifier needs to be built for each class, the program was implemented 10 times to obtain 10 pairs of training and testing data sets. These training and testing data sets were used for classifier training and performance evaluation, respectively.

ML Algorithm Selection and Implementation

A variety of ML algorithms have shown reasonable performance in TC. However, no single algorithm has demonstrated to consistently outperform the others across various applications and domains (Sebastiani 2002). In this research, 10 popular ML algorithms were

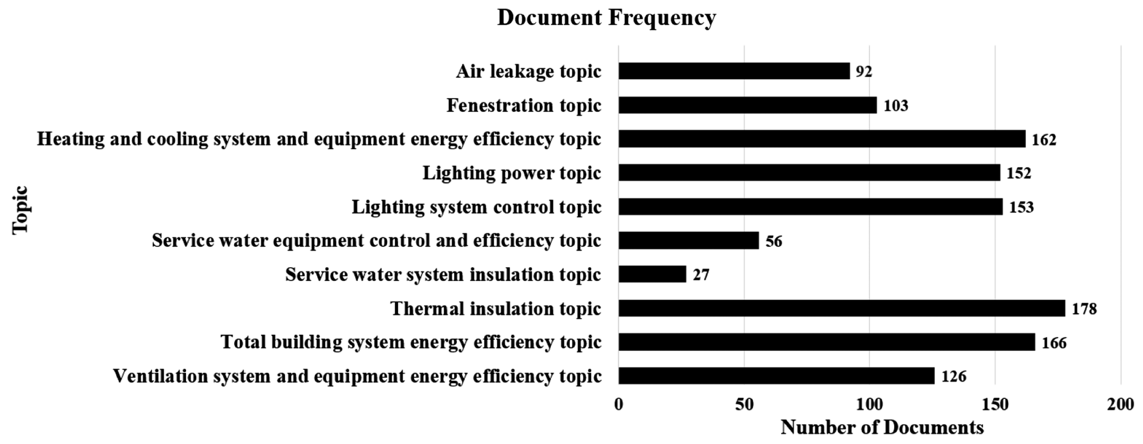

Fig. 4. Document frequency of topics

tested, including SVM (implemented in both linear and rbf kernel), DT (implemented by CART algorithm), NB (implemented by three variances of algorithms: Gaussian NB, multinomial NB, Bernoulli NB), kNN, radius-based neighbors (RBN), nearest centroid (NC), RF, and GBRT (Aggarwal and Zhai 2012; Breiman 2001; Friedman 2001).

Each algorithm has some important parameters that were tuned and optimized by trial and error on the basis of experimental results. Tuning and optimizing parameters refer to the process of looking for the best parameters to maximize the performance. Experimental results refer to the performance yielded when the parameters are tested. For example, parameter C in SVM with linear kernel can control the weight of positive and negative clauses because the number of them could be unbalanced, which may influence the performance of the classifier. To tune the parameter C in SVM, an initial range of values (e.g., 10^{-3}, 10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2}, and 10^{3}) was tested to identify the approximate magnitude of C. Then, a range of specific values (e.g., 10^{-1}–1) in that magnitude was tested to identify the approximately-best C value. The above testing steps were implemented using loops in the Python programming language. The aforementioned ML algorithms were implemented using the Scikit-Learn ML algorithm(s) package written in Python programming language (Pedregosa et al. 2011). The parameters of each algorithm were tuned and optimized to find the closest-to-best parameters that result in the highest performance. Closest-to-best parameters are good enough, because it is infeasible to enumerate all possible values to find the exactly-best parameters (like finding the exact value of π).

Step 4: Classification Result Evaluation

The performance of the aforementioned ML algorithms was evaluated using recall and precision, as per Eqs. (10) and (11), in which true positive (TP) refers to the number of clauses labeled correctly as positive, false positive (FP) refers to the number of clauses labeled incorrectly as positive, and false negative (FN) refers to the number of clauses labeled incorrectly as negative. For this application, recall is more important than precision, because missing to recall one clause means overlooking a relevant clause, which may affect the performance of the ACC system as a whole. Precision is not as critical, because irrelevant text could be filtered out during further IE.

In addition, confusion matrix (CM) was used to analyze the results. CM is a very useful tool to analyze the performance of classifiers (Manning et al. 2009). It is a number-of-classes × number-of-classes matrix, in which the diagonal shows how many testing clauses are labeled correctly, and other positions show how many testing clauses are misclassified from one class to another class. CM can thus help reveal misclassification causes like human errors in labeling

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{10}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{11}$$

Step 5: Performance Improvement

Initial testing and evaluation was conducted without implementing any performance improvement strategies to establish the baseline for comparison. Feature selection and recursive stopword removal were then implemented to explore their effect on improving the performance.

Feature Selection

The effect of two main approaches of feature selection, UFS and RFS, on performance improvement was empirically tested.

As mentioned in the "Background" section, common feature scoring functions used for UFS include CHI, IG, and MI. Although there is no systematical performance difference among these three feature scoring functions, CHI tends to select those more low-frequency features or words (Aggarwal and Zhai 2012; Manning et al. 2009). Because in this application the text is characterized by a relatively small number of features, some low-frequency features may be significant in identifying the class. Therefore, the CHI scoring function was used for testing UFS. After scoring and ranking the features, two types of feature selection methods were used: (1) K-best: a K number of features are selected; and (2) percentage: a certain percentage of features are selected. Fifty to 3,000 features were tested for selecting K, with a 50-feature increasing step size; and 3–75% were tested for selecting the percentage, with a 3% increasing step size.

For testing RFS, the best performing ML algorithm (defined in Step 5, as further discussed in the "Experimental Results and Analysis" section) was used to assign the weights and different combinations of M and N were tested to select the best feature set.

Recursive Domain-Specific Stopword Removal

For each topic, a domain-specific stopword list was created and tested in a recursive manner. As mentioned in Step 3, the standard

English stopword list was initially used to remove those high-frequency but low-content words (e.g., "a" and "the"), which tend to be nondiscriminative for general text. Stopwords are commonly removed using a standard stopword list. However, a domain-specific stopword list might be more descriptive of a specific domain (or subdomain). Domain-specific stopwords are those words that have no discriminative power within a specific domain or context (Makrehci and Kamel 2008). Because construction domain stopword lists are not available, for each topic, a list was created by manually adding domain-specific, nondiscriminative words (such as "include", "allow", and "install") to the original standard English stopword list, in a recursive manner. All words were counted and then high-frequency, low-content words were identified based on domain knowledge and using trial and error.

After removing the stopwords using the general stopword list, the remaining total number of distinct words were approximately 4,000. After sorting these words according to their term frequencies in the whole document collection, it was found that the term frequencies decreased from the levels of 5,000 to 1,000, 1,000 to 500, and 500 to 400 for the first 25, second 25, and third 50 words, respectively. This means that the term frequencies of the remaining 3,900 words were all below 400. Because there is no benchmark to indicate the cutoff term frequency except using trial and error (Manning et al. 2009; Van Rijsbergen 1979), in this paper, those top-100 term frequency words were considered as potential stopwords. These 100 words were then checked and classified into two groups: (1) words that are discriminative of specific topics and thus should be excluded from the stopword list (e.g., "control" is highly related to the lighting control topic, although it appeared more than 3,300 times in the data set), and (2) words that are nondiscriminative of any topic and thus are good potential candidate stopwords (e.g., "include" appeared more than 1,300 times in the data set and is nondiscriminative and nonpredictive of any topic). To determine the final stopwords, these nondiscriminative words (i.e., words in the second group, such as "include", "allow", "according", "foot", "addition", "install", "function", "percent", etc.) were tested one by one for each topic: if removing a word from the features improved performance for a topic, it was added to the domain-specific stopword list of that topic.

Experimental Results and Analysis

The experiments were conducted in a performance-boosting manner; for each step, the technique that yields the best performance was optimized and selected. The final combination of techniques that were selected for all steps forms the best TC algorithm.

Performance of Different of ML Algorithms

The best performance result of each of the 10 tested algorithms for each category is summarized in Table 1. kNN, RF, and SVM showed the top three recall results with 91.60, 91.50, and 89.90% recall values, respectively. They were selected for further comparative evaluation, after implementing feature selection, for the following reasons: (1) RF inherently implements partial feature selection because of its internal algorithm design. So, an apple-to-apple comparison requires further comparison after implementing feature selection for kNN and SVM; and (2) the three algorithms have yielded similar much higher average recall and precision with the least standard deviation, in comparison with the rest of the algorithms. So, there was no need to further evaluate the other algorithms. The results of the comparative evaluation are shown in Table 2. The SVM (with linear kernel) was selected as the optimal algorithm for further performance improvement because it showed relatively robust performance in terms of average and standard deviation of recall and precision. Although the recall of kNN is 0.2% higher than that of SVM, it comes at a high precision cost (more than a 10% reduction in precision).

The relative high performance results of SVM could be explained by the following reasons: (1) SVM is especially suitable for handling environmental regulatory text, because environmental topics can usually be represented by a small set of key, discriminative features (e.g., fenestration topic achieved 100% recall and 82% precision using only 100 features); and (2) these key features usually occur together (e.g., "service," "water," "heating," and "control" occurred together for the service water equipment control and efficiency topic). These properties enable those support vectors to be easily identified for classification, which helps reduce FN errors, thereby improving recall. The relative high performance of SVM with a linear kernel might also indicate that environmental regulatory text does not contain as much ambiguity as other types of text, which would require more complex nonlinear kernels (e.g., rbf, polynomial) for classification.

The use of SVM is also consistent with recent TC research studies in the construction domain, which used SVM, such as in classifying contract documents (Salama and El-Gohary 2013b), project correspondences and meeting minutes (Mahfouz 2011), and safety documents like the U.S. Occupational Safety and Health Administration (OSHA) standards (Chi et al. 2014).

Performance of Different Text Representation Models

As shown in Table 3, the bigram model showed zero precision and recall for half of the topics, indicating that capturing semantic information of environmental regulatory text statistically in terms of word positions in a sentence (as in the bigram model) results in a much decreased performance in comparison to the use of unordered words (as in the BOW model). These results are similar to those reported in other domains and applications (e.g., classifying news articles and medical abstracts) that show that the BOW model could perform better than the bigram model despite the fact that it discards all word association information (Moschitti and Basili 2004). This can be attributed to the following reason: individual relevant features in the BOW model may become irrelevant when associated and combined as new features in the bigram model (Boulis and Ostendorf 2005). Because the relatively small number of features in environmental regulatory text may make the majority of new features in the bigram model unique, during term weighting, these unique features would not contribute to the differentiation of topics. Accordingly, the BOW model was empirically selected for text representation.

Performance of Different Term Weighting Schemes

The TFIDF, $TFRF_M$, and $TF_{max}RF_M$ weighting schemes were tested using the previously selected BOW model and SVM algorithm. The performance results are shown in Table 4. To ensure that the performance of different weighting schemes is not affected by feature selection, comparative experiments were conducted for all weighting schemes both with and without feature selection. Although different weighting schemes show different performances for different topics, only one single term weighting scheme must be selected for all topics to avoid the use of multiple term weighting schemes in one classifier. Thus, the average recall of all topics and the corresponding standard deviation (SD) were used for weighting scheme selection. Two selection criteria were used: (1) highest recall and (2) lowest SD, which indicates robust performance across topics. Before feature selection, TFIDF achieved the best average

Table 1. Performance of Different ML Algorithms (before Feature Selection)

	Performance of ML algorithm																			
	SVM		DT		Gaussian NB		Multinomial NB		Bernoulli NB		kNN		RBN		NC		RF		GBRT	
Topic	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)	Recall (%)
Air leakage topic	87	90	85	76	70	72	95	62	86	62	85	90	96	38	84	91	89	81	77	85
Fenestration topic	84	90	89	72	72	83	85	86	81	60	84	91	96	43	85	86	89	93	91	90
Heating and cooling system and equipment energy efficiency topic	87	89	39	83	82	77	80	82	73	83	84	83	83	11	79	87	82	85	92	75
Lighting power topic	89	94	34	96	78	86	79	96	86	81	85	95	80	40	89	89	87	96	80	93
Lighting system control topic	82	95	65	95	80	84	75	95	82	91	79	96	86	47	79	91	74	99	80	95
Service water equipment control and efficiency topic	74	80	64	84	58	60	64	36	75	12	71	88	75	12	71	80	82	92	82	72
Service water system insulation topic	76	100	71	92	67	77	0	0	75	23	68	100	100	46	69	85	92	92	71	92
Thermal insulation topic	98	89	51	93	88	80	59	97	31	98	92	94	21	100	90	90	84	97	90	93
Total building system energy efficiency topic	91	89	62	67	76	74	66	91	81	75	90	92	82	18	85	88	90	91	79	84
Ventilation system and equipment energy efficiency topic	91	83	84	76	79	74	91	81	93	69	92	87	100	20	89	81	91	89	91	88
Average	85.9	89.9	64.4	83.4	75.0	76.7	69.4	72.6	76.3	65.4	83.0	91.6	81.9	37.5	82.0	86.8	86.0	91.5	83.3	86.7
Standard deviation	7.22	5.70	18.89	10.37	8.54	7.50	27.01	31.70	17.03	28.02	8.21	4.93	23.17	26.13	7.39	3.88	5.54	5.46	7.24	7.83

Table 2. Performance of Different ML Algorithms (after Feature Selection)

	Performance of ML algorithm					
	SVM		kNN		RF	
Topic	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)	Recall (%)
Air leakage topic	86	93	71	88	89	81
Fenestration topic	84	93	90	93	89	93
Heating and cooling system and equipment energy efficiency topic	86	90	82	86	82	85
Lighting power topic	85	97	44	100	87	96
Lighting system control topic	84	97	54	99	74	99
Service water equipment control and efficiency topic	77	96	82	92	82	92
Service water system insulation topic	76	100	100	100	92	92
Thermal insulation topic	95	94	75	98	84	97
Total building system energy efficiency topic	88	92	56	97	90	91
Ventilation system and equipment energy efficiency topic	91	83	94	84	91	89
Average	85.2	93.5	74.8	93.7	86.0	91.5
Standard deviation	5.71	4.70	18.52	6.02	5.54	5.46

Table 3. Performance of Different Text Representation Models

	Performance of text representation model (no feature selection, SVM with linear kernel)					
	BOW			Bigram		
Topic	C[a]	Precision (%)	Recall (%)	C[a]	Precision (%)	Recall (%)
Air leakage topic	1	87	90	0.4	26	13
Fenestration topic	0.8	84	90	150	0	0
Heating and cooling system and equipment energy efficiency topic	2	87	89	0.1	0	0
Lighting power topic	1	89	94	0.3	28	48
Lighting system control topic	0.7	82	95	0.1	0	0
Service water equipment control and efficiency topic	2	74	80	0.2	0	0
Service water system insulation topic	1	76	100	0.1	0	0
Thermal insulation topic	3	98	89	9	13	31
Total building system energy efficiency topic	2	91	89	9	15	12
Ventilation system and equipment energy efficiency topic	0.7	91	83	40	17	2

[a]C is a penalty parameter used in SVM that adjusts the data unbalance problem.

recall of 89.9% with the least SD of 5.7%, compared with 89.0 and 85.3% recall and 6.88 and 8.25% SD for $TF_{max}RF_M$ and $TFRF_M$, respectively. After feature selection, TFIDF still outperformed in terms of recall (93.5%) but without achieving the least SD (4.70%). $TF_{max}RF_M$ achieved 92.1% recall and lowest SD of 3.96%, whereas $TFRF_M$ still yielded the lowest recall of 91.1% and highest SD of 5.59%. Accordingly, TFIDF was selected as the optimal term weighting scheme because of the desired high recall.

Additionally, the following three observations were made. First, $TF_{max}RF_M$ consistently outperformed $TFRF_M$ in terms of both the average and the standard deviation of recall and precision. These findings are similar to those reported in the news domain (as tested on the 20 Newsgroups and Reuters News data sets) (Xuan and Quang 2014). Second, $TF_{max}RF_M$ outperformed TFIDF in terms of precision. Not only did it yield the best precision at 7 out of the 10 topics, it also yielded the best average precision among the three weighting schemes. Third, feature selection did not affect the recall ordering of the three weighting schemes.

Baseline Performance

Before the implementation of performance improvement strategies, the initially selected combination of techniques (BOW model, TFIDF weighting, and SVM algorithm with linear kernel) was used as a baseline for comparison. The results before and after implementing the improvement strategies are summarized in Table 5. Table 5 shows the corresponding best parameters found when the highest recall is achieved. For the baseline condition, an average performance of 89.9 and 85.9% recall and precision, respectively, was achieved.

Effect of Feature Selection

The CHI scoring function and the K-best feature selection method were empirically selected as the optimal methods for feature selection, because based on the experimental results they together outperformed RFS as well as CHI and percentage feature selection. Based on the results, feature selection has been shown to be effective in improving the performance in terms of average recall (Table 5). The average recall and precision have reached 93.5 and 85.2%, respectively. The results also demonstrate the expected trend that an increase in recall (3.6% in this case) decreases precision (0.7% in this case). The highest improvement was achieved for the service water equipment control and efficiency topic at a 16% increase in recall (reaching 96% recall) using 550 features. Only one of the 10 topics did not show any improvement in recall (ventilation system and equipment energy efficiency topic), which indicates that feature selection does not necessarily improve recall for all classes.

Table 4. Performance of Different Term Weighting Schemes (before and after Feature Selection)

	Performance of term weighting scheme											
	TFIDF				$TF_{max}RF_M$				$TFRF_M$			
	Before FS[a]		After FS[a]		Before FS[a]		After FS[a]		Before FS[a]		After FS[a]	
Topic	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)	Recall (%)
Air leakage topic	87	90	86	93	97	97	97	97	89	78	90	84
Fenestration topic	84	90	84	93	95	86	91	93	93	87	96	96
Heating and cooling system and equipment energy efficiency topic	87	89	86	90	77	86	79	88	83	80	83	83
Lighting power topic	89	94	85	97	89	91	89	93	87	83	87	85
Lighting system control topic	82	95	84	97	90	83	92	85	96	92	96	92
Service water equipment control and efficiency topic	74	80	77	96	75	75	78	90	75	71	76	94
Service water system insulation topic	76	100	76	100	82	90	82	90	58	100	64	100
Thermal insulation topic	98	89	95	94	88	98	91	98	88	89	90	91
Total building system energy efficiency topic	91	89	88	92	88	90	92	93	91	91	91	91
Ventilation system and equipment energy efficiency topic	91	83	91	83	94	94	94	94	80	82	84	95
Average	85.9	89.9	85.2	93.5	87.5	89.0	88.5	92.1	84.0	85.3	85.7	91.1
Standard deviation	7.22	5.70	5.71	4.70	7.41	6.88	6.52	3.96	11.05	8.25	9.74	5.59

[a]FS = feature selection.

Table 5. Effects of Feature Selection and Recursive Domain-Specific Stopword Removal on Performance

	Performance before and after using feature selection and domain-specific stopword removal													
	No feature selection, SVM with linear kernel			Feature selection using K-best, SVM with linear kernel					Feature selection using K-best, domain-specific stopword removal, SVM with linear kernel					
Topic	C[a]	Precision (%)	Recall (%)	C[a]	K[b]	Precision (%)	Recall (%)	Increase in recall[c] (%)	C[a]	K[b]	Stopwords[d]	Precision (%)	Recall (%)	Increase in recall[c] (%)
Air leakage topic	1	87	90	2	750	86	93	+3	4	700	Allow, include	82	98	+5
Fenestration topic	0.8	84	90	9	350	84	93	+3	6	100	Allow, include, according	82	100	+7
Heating and cooling system and equipment energy efficiency topic	2	87	89	2	2,050	86	90	+1	2	1,100	Allow, include, section, area, multiply	83	97	+7
Lighting power topic	1	89	94	5	1,150	85	97	+3	5	1,150	N/A	85	97	+0
Lighting system control topic	0.7	82	95	2	2,850	84	97	+2	2	2,850	N/A	84	97	+0
Service water equipment control and efficiency topic	2	74	80	9	550	77	96	+16	9	550	N/A	77	96	+0
Service water system insulation topic	1	76	100	7	650	76	100	+0	5	700	According, addition	88	100	+0
Thermal insulation topic	3	98	89	6	850	95	94	+5	6	850	N/A	95	94	+0
Total building system energy efficiency topic	2	91	89	3	1,000	88	92	+3	7	550	Include, install, foot	83	97	+5
Ventilation system and equipment energy efficiency topic	0.7	91	83	0.7	2,500	91	83	+0	2	1,450	Include, allow, percent	84	97	+14
Average	—	85.9	89.9	—	—	85.2	93.5	+3.6	—	—	—	84.3	97.3	+3.8
Standard deviation	—	7.22	5.70	—	—	5.71	4.70	—	—	—	—	4.67	1.77	—

[a]C = penalty parameter used in SVM that adjusts the data unbalance problem.
[b]K = the number of features selected to achieve the best performance, at a unit of 50.
[c]Shows percentage increase in recall compared with previous performance.
[d]N/A means there are no effective stopwords found to improve performance.

The number of features selected at the highest recall (K value shown in Table 5) provides some insight about the differences across classes. For example, the fenestration topic used 350 features only to achieve maximum recall, which indicates that clauses belonging to the fenestration topic can be easily classified. In contrast, the ventilation system and equipment energy efficiency topic used 2,500 features but still gained no improvement in recall, which indicates that clauses belonging to this topic are harder to differentiate. Similarly, other topics using a relatively large K value (relative to other topics in this data set) to achieve best recall showed less recall improvement. For example, the heating and cooling system and equipment energy efficiency topic used 2,050 features but achieved only 1% recall increase. In addition, a relatively large K value may also imply that potential subtopics may exist for that topic, thereby needing to aggregate more features from each subtopic to better represent their parent topic. Overall, none of the 10 topics used more than 2,900 features to achieve its best recall. This shows the effectiveness of feature selection in enhancing recall, even if the original feature size is relatively small.

Effect of Recursive Domain-Specific Stopword Removal

The performance was significantly improved after using the proposed domain-specific stopword lists (Table 5). This indicates that the use of domain-specific text characteristics is effective in improving the performance of classification. The final performance shows an average 97.3 and 84.3% recall and precision, respectively. The average recall increased by 3.8% at the expense of a 0.9% decrease in precision. The standard deviation of both recall and precision continued to decrease and finally dropped to 1.77 and 4.67%, respectively, which indicates that the proposed TC algorithm is relatively robust on all topics. The results also show the following two findings. First, a change of stopwords caused a variation in the best parameters. Using parameter K as an illustration, all those topics that achieved improvement in terms of recall used fewer features (e.g., the ventilation system and equipment energy efficiency topic used 1,450 features instead of 2,500 features to gain 14% increase in recall), whereas topics that gained improvement in terms of precision used more features (e.g., the service water system insulation topic used 50 more features to reach 12% increase in precision meanwhile still maintaining 100% recall). This observation may also substantiate the counteractive recall-precision relationship in the aspect of number of features: selecting more features may help identify more feature differences among topics which reduces FP errors, but may increase FN errors thereby undermining the recall, and vice versa. Second, stopwords varied across classes. This indicates that different topics (and subtopics) may require different stopword lists.

Contribution to the Body of Knowledge

This work offers a domain-specific, ML-based hierarchical TC algorithm for classifying clauses in environmental regulatory documents according to a semantic TC topic hierarchy. The topic hierarchy is part of a deontic ACC model. This algorithm is key in enabling automated environmental compliance checking in the construction domain by enhancing the efficiency of automated IE. In comparison to the authors' previous efforts in TC for ACC in construction (Salama and El-Gohary 2013b), this algorithm addresses a more challenging TC problem—hierarchical TC as opposed to nonhierarchical TC. Hierarchical TC allows for a more granular classification of text according to detailed subtopics (e.g., "thermal insulation" as opposed to "environmental") and thus would result in further enhancement of automated IE efficiency.

Beyond this application, this work additionally contributes to the body of knowledge, in seven main ways. First, it offers a baseline domain-specific, ML-based hierarchical TC algorithm for classifying environmental regulatory text according to a hierarchy of topics. Future research efforts could use this work as a benchmark and could adapt the algorithm to classify other types of environmental documents (e.g., EPA regulations) and using other types of topic hierarchies (e.g., hierarchy of environmental emergencies like chemical pollution). Second, it shows that high recall and precision results can be achieved for a relatively deep TC granularity level (classifying the text according to topics in the fifth level of the hierarchy) for environmental regulatory text, and that the flat approach in dealing with hierarchical TC is effective. As you go to a more specialized level of topics, TC becomes typically more challenging (Khan et al. 2014) and performance could highly drop [e.g., in the construction domain, classification accuracy dropped from 95.88% at the first level to 86.37% at the third level of the hierarchy (Caldas and Soibelman 2003)]. Compared with the authors' previous work in nonhierarchical TC (Salama and El-Gohary 2013b), this study shows a relatively small drop in recall, a drop from 100% at the first level to 97% at the fifth level. Third, the research shows the effectiveness of adopting a multiclass classification approach to deal with multilabel classification problems in avoiding a data imbalance problem. The use of a multiclass classification approach is, thus, especially helpful in cases in which the document frequency is imbalanced across different topics. Fourth, this research offers two modified supervised term weighting schemes, which were adapted to multiclass classification problems. The experimental results showed that they did not perform as well, in this application, compared with the commonly used TFIDF weighting scheme. But, having those modified weighting schemes would allow other researchers to further evaluate them in classifying other types of documents (e.g., OSHA standards) and in other domains (e.g., safety). Fifth, this research shows the effectiveness of feature selection in enhancing recall, even if the original feature size is relatively small (approximately 4,200 features compared with the millions of features that are commonly seen in CS domain data sets). It shows that a small selected feature size (less than 2,900) is enough to achieve high performance. Sixth, the research shows that recursive, domain-specific stopword removal is very effective in improving recall. It shows that the use of a general stopword list is not sufficient as the domain of text becomes highly specialized, and that the development and use of domain-specific stopword lists is highly effective in achieving increased performance at a low manual effort, especially that such lists are reusable. Seventh, this work offers a data set of labeled clauses for the energy efficiency subdomain. Unlike other domains, there is a lack of benchmark data sets in the construction domain (like the 20 Newsgroups data set) that could be used for TC performance evaluation. Benchmark data sets are important for research, because they are used by researchers for comparative evaluation of different techniques or algorithms. For example, with such a benchmark data set, different TC algorithms in the construction domain could be tested on the same data set, and their performance could be evaluated relative to a known benchmark.

Conclusions and Future Work

This paper presented a domain-specific, ML-based hierarchical TC algorithm for classifying clauses in environmental regulatory documents into a number of hierarchically detailed topics for supporting

ACC in the construction domain. The algorithm classifies clauses according to leaf topics at the fifth level of a semantic TC topic hierarchy. The algorithm, thus, addresses a relatively deep TC granularity level, and therefore a more challenging TC problem. As you go to a more specialized level of topics, TC becomes typically more challenging, because the levels of knowledge and terminology of the text become more specialized and more specific, which make the text harder to discriminate. A flat approach was used to deal with the hierarchical TC problem. The multilabel classification problem was transformed into a multiclass classification problem. For preparing the training and testing data, approximately 1,200 clauses were collected from 10 environmental regulatory documents, such as the 2012 International Energy Conservation Code (ICC 2012), and were classified into 10 leaf subtopics of the energy efficiency topic (a subclass of environmental topic in the semantic topic hierarchy).

In developing the TC algorithm, the following techniques were tested and evaluated in terms of average recall and precision and their standard deviation: (1) 10 popular ML algorithms; (2) two text representation methods (BOW model and bigram model); and (3) three term weighting schemes—two supervised term weighting schemes ($TFRF_M$ and $TF_{max}RF_M$) that were modified to adapt them to multiclass classification and one unsupervised term weighting scheme (TFIDF) that is commonly used. The best performance was achieved using SVM algorithm with linear kernel, BOW model, and TFIDF weighting.

For further performance enhancement, two performance improvement strategies were implemented: (1) feature selection (a number of methods were tested and, accordingly, K-best feature selection method and CHI feature scoring function were selected) and (2) domain-specific stopword removal (construction-domain-specific stopword lists were created and used to facilitate domain adaptation). A number of primary conclusions were drawn during performance improvement: (1) both strategies were effective in enhancing recall; (2) during this process, the standard deviation of both recall and precision continued to decrease, which indicates enhanced performance consistency and robustness; and (3) each environmental topic (or subtopic) may need a different stopword list. The final classifier achieved approximately 97 and 84% average recall and precision, respectively, on the testing data. Two characteristics of environmental regulatory text may have contributed to such performance. Compared with general text like that in news articles, environmental regulatory text is (1) more specialized in terms of topics; specialized text is usually characterized by a smaller number of features and, thus, more discriminative features; and (2) more standardized in terms of terminology, with less homonyms and synonyms; standardized terminology results in higher term frequencies and, thus, more discriminative features.

In future work, the authors will (1) explore the use (or adaptation) of the proposed TC algorithm for classifying other types of regulatory documents (e.g., safety regulatory documents such as OSHA standards) on the basis of other types of topics (e.g., safety topics); (2) explore the reusability of the developed stopword lists in classifying other types of environmental documents and automate the process of constructing domain-specific stopword lists; (3) explore the use of other approaches for domain adaptation, in addition to domain-specific stopword lists such as feature augmentation; and (4) explore the use of ontologies and development of an ontology-based TC algorithm to take advantage of the semantic information of the text to further improve the classification performance, and compare the ontology-based approach with the ML-based approach.

Acknowledgments

The authors would like to thank the National Science Foundation (NSF). This material is based upon work supported by NSF under Grant No. 1201170. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF.

References

Aggarwal, C., and Zhai, C. (2012). *A survey of text classification algorithms*, Springer, New York, 163–222.

Aly, M. (2005). "Survey on multiclass classification methods." Caltech, Pasadena, CA, ⟨http://www.vision.caltech.edu/malaa/publications/aly05multiclass.pdf⟩ (Feb. 25, 2015).

ASHRAE (American Society of Heating, Refrigerating, and Air-Conditioning Engineers). (2010). "ANSI/ASHRAE/IES standard 90.1-2010 energy standard for buildings except low-rise residential buildings." ⟨ftp://law.resource.org/pub/us/code/ibr/ashrae.90.1.ip.2010.pdf⟩ (Jul. 30, 2015).

Blitzer, J., Dredze, M., and Pereira, F. (2007). "Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification." *Proc., 42nd Annual Meeting on Association for Computational Linguistics*, Association for Computational Linguistics, Stroudsburg, PA, 440–447.

Boulis, C., and Ostendorf, M. (2005). "Text classification by augmenting the bag-of-words representation with redundancy-compensated bigrams." *Proc., Int. Workshop Feature Selection in Data Mining*, International Computer Science Institute, Berkeley, CA, 9–16.

Breiman, L. (2001). "Random forests." *Mach. Learn.*, 45(1), 5–32.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and regression trees*, Chapman & Hall/CRC, Boca Raton, FL.

Caldas, C. H., and Soibelman, L. (2003). "Automating hierarchical document classification for construction management information systems." *Automat. Constr.*, 12(4), 395–406.

Caldas, C. H., Soibelman, L., and Han, J. (2002). "Automated classification of construction project documents." *J. Comput. Civ. Eng.*, 10.1061/(ASCE)0887-3801(2002)16:4(234), 234–243.

Chang, C., and Lin, C. (2011). "LIBSVM: A library for support vector machines." *ACM Trans. Intell. Syst. Technol.*, 2(3), 1–27.

Chi, N. W., Lin, K. Y., and Hsieh, S. H. (2014). "Using ontology-based text classification to assist job hazard analysis." *Adv. Eng. Inform.*, 28(4), 381–394.

Eastman, C., Lee, J., Jeong, Y., and Lee, J. (2009). "Automatic rule-based checking of building designs." *Automat. Constr.*, 18(8), 1011–1033.

El-Gohary, N., and El-Diraby, T. (2010). "Domain ontology for processes in infrastructure and construction." *J. Constr. Eng. Manage.* 10.1061/(ASCE)CO.1943-7862.0000178, 730–744.

Fagni, T., and Sebastiani, F. (2010). "Selecting negative examples for hierarchical text classification: An experimental comparison." *J. Am. Soc. Inform. Sci. Technol.*, 61(11), 2256–2265.

Friedman, J. H. (2001). "Greedy function approximation: A gradient boosting machine." *Ann. Stat.*, 29(5), 1189–1232.

Ghamrawi, N., and McCallum, A. (2005). "Collective multilabel classification." *Proc., 14th ACM Int. Conf. on Information and Knowledge Management*, Association for Computing Machinery (ACM), New York, 195–200.

Giorgetti, D., and Sebastiani, F. (2003). "Multiclass text categorization for automated survey coding." *Proc., 2003 ACM Symp. on Applied Computing*, Association for Computing Machinery (ACM), New York, 798–802.

ICC (International Code Council). (2012). "2012 international energy conservation code." ⟨http://publicecodes.cyberregs.com/icod/iecc/2012/⟩ (Jul. 30, 2015).

Khan, S., Baig, A. R., and Shahzad, W. (2014). "A novel ant colony optimization based single path hierarchical classification algorithm for predicting gene ontology." *Appl. Soft Comput.*, 16, 34–49.

Kovacevic, M., Nie, J. Y., and Davidson, C. (2008). "Providing answers to questions from automatically collected web pages for intelligent decision making in the construction sector." *J. Comput. Civ. Eng.*, 10.1061/(ASCE)0887-3801(2008)22:1(3), 3–13.

Liao, C., Alpha, S., and Dixon, P. (2003). "Feature preparation in text categorization." *Proc., Autralasian Data Mining: Workshop*, Springer, Berlin.

Mahfouz, T. (2011). "Unstructured construction document classification model through support vector machine (SVM)." *Proc., 2011 ASCE Int. Workshop on Computing in Civil Engineering*, ASCE, Reston, VA, 126–133.

Makrehchi, M., and Kamel, M. S. (2008). "Automatic extraction of domain-specific stopwords from labelled documents." *Proc., Information Retrieval Research, 30th European Conf. on Advance Information Retrieval*, Springer, Berlin, 222–233.

Malkani, Z., and Gillie, Z. (2012). "Supervised multiclass classication of tweets." Stanford Univ., Stanford, CA, ⟨http://cs229.stanford.edu/proj2012/GillieMalkani-SupervisedMulticlassClassificationOfTweets.pdf⟩ (Mar. 4, 2015).

Man, L., Tan, C. L., Jian, S., and Yue, L. (2009). "Supervised and traditional term weighting methods for automatic text categorization." *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(4), 721–735.

Manning, C., Raghavan, P., and Shutze, H. (2009). *Introduction to information retrieval*, Cambridge University Press, Cambridge, U.K.

Manning, C. D., and Schutze, H. (1999). *Foundations of statistical natural language processing*, MIT Press, Cambridge, MA.

Moschitti, A., and Basili, R. (2004). "Complex linguistic features for text classification: A comprehensive study." *Advance information retrieval*, Springer, Berlin, 181–196.

Nawari, N. (2012). "Automating codes conformance." *J. Archit. Eng.*, 10.1061/(ASCE)AE.1943-5568.0000049, 315–323.

Pedregosa, F., et al. (2011). "Scikit-learn: Machine learning in python." *J. Mach. Learn. Res.*, 12(10), 2825–2830.

Porter, M. F. (2006). "The English (Porter2) stemming algorithm." ⟨http://snowball.tartarus.org/algorithms/english/stemmer.html⟩ (Dec. 17, 2014).

Salama, D., and El-Gohary, N. (2013a). "Automated compliance checking of construction operation plans using a deontology for the construction domain." *J. Comput. Civ. Eng.*, 10.1061/(ASCE)CP.1943-5487.0000298, 681–698.

Salama, D., and El-Gohary, N. (2013b). "Semantic text classification for supporting automated compliance checking in construction." *J. Comput. Civ. Eng.*, 10.1061/(ASCE)CP.1943-5487.0000301, 04014106.

Sebastiani, F. (2002). "Machine learning in automated text categorization." *J. ACM Comput. Surv.*, 34(1), 1–47.

Silla, C., Jr., and Freitas, A. (2011). "A survey of hierarchical classification across different application domains." *Data Min. Knowl. Discovery*, 22(1-2), 31–72.

Sun, A., and Lim, E. (2001). "Hierarchical text classification and evaluation." *Proc., 2001 IEEE Int. Conf. on Data Mining*, IEEE Computer Society, Washington, DC, 521–528.

Sun, A., Lim, E., and Ng, W. (2003). "Performance measurement framework for hierarchical text classification." *J. Am. Soc. Inform. Sci. Technol.*, 54(11), 1014–1028.

Sun, Y. M., Kamel, M. S., Wong, A. K. C., and Wang, Y. (2007). "Cost-sensitive boosting for classification of imbalanced data." *Pattern Recogn.*, 40(12), 3358–3378.

Tan, X., Hammad, A., and Fazio, P. (2010). "Automated code compliance checking for building envelope design." *J. Comput. Civ. Eng.*, 10.1061/(ASCE)0887-3801(2010)24:2(203), 203–211.

Tsoumakas, G., and Katakis, I. (2007). "Multilabel classification: An overview." *Int. J. Data Warehouse Min.*, 3(3), 1–13.

Van Rijsbergen, C. J. (1979). *Information retrieval*, Butterworth-Heinemann, Newton, MA.

Wu, K., Lu, B. L., Uchiyama, M., and Isahara, H. (2007). "A probabilistic approach to feature selection for multiclass text categorization." *Advance in neural networks*, Springer, Berlin, 1310–1317.

Xuan, N., and Quang, H. (2014). "A new improved term weighting scheme for text categorization." *Knowledge and systems engineering*, Springer, Cham, Switzerland, 261–270.

Yoon, Y.Lee, C., and Lee, G. G. (2006). "An effective procedure for constructing a hierarchical text classification system." *J. Am. Soc. Inform. Sci. Technol.*, 57(3), 431–442.

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.

Zhong, B. T., Ding, L. Y., Luo, H. B., Zhou, Y., Hu, Y. Z., and Hu, H. M. (2012). "Ontology-based semantic modeling of regulation constraint for automated construction quality compliance checking." *Autom. Constr.*, 28(2012), 58–70.

Zhou, P., and El-Gohary, N. (2014). "Semantic-based text classification of environmental regulatory documents for supporting automated environmental compliance checking in construction." *Proc., 2014 ASCE Construction Research Congress Congress (CRC)*, ASCE, Reston, VA.