Final published version is found in the ASCE Library here:
https://doi.org/10.1061/(ASCE)CP.1943-5487.0000917
Xue, X., and Zhang, J. (2020). "Building codes Part-of-Speech tagging performance improvement by error-driven transformational rules." J. Comput. in Civ. Eng., 34(5), 04020035.

Building Codes Part-of-Speech Tagging Performance Improvement by Error-Driven Transformational Rules

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

Abstract

To enable full automation, automated code compliance checking systems need to extract regulatory information in building codes and convert it to computable representations. This conversion is a natural language processing (NLP) task that requires highly accurate part-of-speech (POS) tagging results on building codes. Existing POS taggers, however, do not provide such accuracy on building codes. To address this need, the authors propose to improve the performance of POS taggers by error-driven transformational rules that revise machine tagged POS results. The proposed method utilizes a syntactic and semantic rule-based, NLP approach combined with a structure that is inspired by transfer learning. This method generates a group of transformational rulesets, from simple ones to complex ones, that will convert machine taggers' tagging results to their corresponding human-labeled gold standard. The transformational rules utilize syntactic and semantic information of domain texts. All rules are constrained not to introduce any errors when fixing existing errors of machine taggers. The last ruleset, which fixes most common remaining errors in textual data after all other rules are applied, is exempted from this constraint. An experimental testing on Part-of-Speech Tagged Building Codes (PTBC) data shows this method reduced 78.91% of errors in POS tagging results of building codes, which increased the POS tagging accuracy on building codes from 89.13% to 98.12%.

Civil Engineering (CE) Database Subject Headings: Project management; Construction management; Information management; Computer applications; Artificial intelligence.

[1] Automation and Intelligent Construction (AutoIC) Lab, School of Construction Management Technology, Purdue University, West Lafayette, IN, 47907, PH (765) 430-2009. email: xue39@purdue.edu.
[2] Automation and Intelligent Construction (AutoIC) Lab, School of Construction Management Technology, Purdue University, West Lafayette, IN, 47907, PH (765) 494-1574; FAX (765) 496-2246. (corresponding author). email: zhan3062@purdue.edu.

Final published version is found in the ASCE Library here:
https://doi.org/10.1061/(ASCE)CP.1943-5487.0000917
Xue, X., and Zhang, J. (2020). “Building codes Part-of-Speech tagging performance improvement by error-driven transformational rules.” J. Comput. in Civ. Eng., 34(5), 04020035.

Author Keywords: Automated compliance checking; Automated information extraction; Natural language processing; Part-of-speech tagging; Automated construction management systems.

Introduction

Traditional manual building code compliance checking methods have several limitations, such as (1) a lengthy review process, (2) a high cost, and (3) error-prone results (Alghamdi et al. 2017; Lee et al. 2018; Preidel and Borrmann 2017; Sacks et al. 2019). It remains a paper-based, manual, and non-standardized process that requires constant human attention and inputs. A traditional building plan review process begins with the building permit applicant submitting a range of hard copy documents, including all the drawings, specifications, documentations and contracts of a project, and a plan review fee, to a building authority. Any mistakes or omissions in the submitted building plans will cause the submission returned to the applicant with requests for revision or additional information. The applicant needs to respond to this request within a certain time frame. The building authority reviews the response and checks the updated building plans. This process may last several weeks or several months until the building authority issues a building permit (City of Savannah 2019). For example, in San Clemente, California, this process can last 120 days for accessory dwelling units (City of San Clemente 2019). Not limited to the building itself, many sub-systems or components of a building require sperate reviews and permissions, such as heating, ventilation, and air conditioning (HVAC) systems (Lopes et al. 2011), fire alarm systems (City of El Cajon Community Development Department 2019), and elevators (State of California 2016), which may further increase the cost and time to obtain building permits. The time and cost needed to obtain building permits also increase with the complexity of modern construction projects (City of Chicago 2019). At the same time, local governments’ diverse adaptation of buildings codes (Ching and Winkel 2018) further increases complexities in code compliance. The demand of innovative code compliance checking systems rises.

Final published version is found in the ASCE Library here:
https://doi.org/10.1061/(ASCE)CP.1943-5487.0000917
Xue, X., and Zhang, J. (2020). "Building codes Part-of-Speech tagging performance improvement by error-driven transformational rules." J. Comput. in Civ. Eng., 34(5), 04020035.

Background

Automated Code Compliance Checking

To address the increasing demand in building permits, many researchers and industry experts introduced new methods of code compliance checking. Their efforts focus on making code compliance checking paperless, automated and standardized. The structural design checking using decision table (Fenves 1966) was one of the first efforts in this domain (İlal and Günaydın 2017), which led to many attempts to create expert systems for building codes in the 1980s (Dimyadi and Amor 2013), such as the Standard Interface for Computer Aided Design (SICAD) (Lopez et al. 1989), the Standards Processing Expert (SPEX) (Delis and Delis 1995; Garrett and Fenves 1987), and the Design Prototypes (Gero 1990). However, low performance and high maintenance cost of expert systems in the 1980s limited these attempts only to proofs of concepts with a lack of actual implementations. An expert system, which uses a vast body of domain-specific knowledge stored in a computer (Liao 2005), has limitations such as high maintenance cost, difficulty in scalability, and the narrow range of applications (Chollet 2017). These forerunners' efforts gave birth to more recent code compliance checking expert systems, such as BCAider and DesignCheck, in early 2000s (Dimyadi and Amor 2013). In addition, there were expert systems that focused on building codes in a specific domain or a limited range of domains in 1980s and 1990s. For example, the Fire-Code Analyzer (Delis and Delis 1995) focused on fire protection related codes in New Zealand, the Life Safety Code Advisor focused on National Fire Protection Association (NFPA) safety code in the U.S., and the TALLEX (Sabouni and Al-Mourad 1997) focused on tall buildings in the United Arab Emirates (UAE).

In the 2000s, building information modeling (BIM) dramatically changed the way code compliance systems work by providing a reliable digital representation of buildings (Nguyen and Kim 2011). For example, Solibri Model Checker (SMC) started as a BIM validation tool, and it obtained code compliance checking ability in its later updates (Eastman et al. 2009). Singapore government initiated the Construction and Real

Final published version is found in the ASCE Library here:
https://doi.org/10.1061/(ASCE)CP.1943-5487.0000917
Xue, X., and Zhang, J. (2020). "Building codes Part-of-Speech tagging performance improvement by error-driven transformational rules." J. Comput. in Civ. Eng., 34(5), 04020035.

Estate Network (CORENET) project, which allows BIM model, instead of papers, to be submitted for plan review. The UK government started to require submissions of BIM for all public projects that are funded by the British Central Government from 2016 (UK BIM Task Group 2016). The KBimCode in South Korea was capable of code compliance checking of BIM against building codes, but it needs manual efforts to convert building codes from natural language to a computer-processable format (Choi and Kim 2017).

Gap in Existing Information Extraction Systems

With BIM as a reliable digital representation of buildings, code compliance checking systems made great progress over the last two decades. However, they are still far from a wide real-world deployment. In many current automated code compliance systems, information extraction and information transformation rely on domain experts' manual efforts to convert building codes to a computer-processable format, such as decision tables (Tan et al. 2010), regulatory knowledge model (Dimyadi et al. 2016), and structured regulatory information rulesets (İlal and Günaydın 2017).

Based on existing literature, current code compliance checking systems lack automated regulatory information extraction and transformation capabilities. By drafting building codes in computer-checkable logic clauses or rulesets instead of natural language, code compliance checking systems can bypass the needed information extraction and transformation step and achieve full automation in an alternative way. However, such a dramatic shift is not expected in a foreseeable future (Bell et al. 2009; Li et al. 2012). In addition, the large size of existing building codes creates further challenges in achieving such a transition. In the U.S., local jurisdictions usually apply customizations and modifications to standard codes published by the international code council (ICC), which further contribute to the complexity of the body of building codes. Automated information extraction and transformation are necessary for automated code compliance systems to function on existing as well as forthcoming building code versions. Some researchers proposed sematic analysis of building codes through deep learning for information extraction, but the extracted

Final published version is found in the ASCE Library here:
https://doi.org/10.1061/(ASCE)CP.1943-5487.0000917
Xue, X., and Zhang, J. (2020). "Building codes Part-of-Speech tagging performance improvement by error-driven transformational rules." J. Comput. in Civ. Eng., 34(5), 04020035.

information failed to convert to checkable rules (Song et al. 2018). Pattern matching-based natural language processing method, on the other hand, can generate logic clauses through information extraction and transformation with a high accuracy (Zhang and El-Gohary 2015; Li et al. 2016; Xu and Cai 2020). The pattern matching-based method of Zhang and El-Gohary (2015) can convert natural language provisions to logic clauses, and their entire automated code compliance checking method reached a 98.7% recall and 87.6% precision in non-compliance detection (Zhang and El-Gohary 2017). However, to enable real-world applications, the recall must be improved to 100%. The main sources of errors reported by Zhang and El-Gohary (2017) were of two types: limitations of the extraction and transformation rules, and limitations of the state-of-the-art POS taggers' performance on building codes. Reducing/eliminating such errors were expected to further improve the overall non-compliance detection performance. In this paper, the authors focus on addressing the performance of the state-of-the-art POS taggers on building codes, because the extraction and transformation rules use the POS tagging information and therefore rely on its performance.

Part-of-Speech Tagging

A fully automated code compliance checking system could be an NLP-based system with an essential information extraction and transformation component. The information extraction and transformation component utilizes part-of-speech information as well as other syntactic/semantic information of building codes provisional sentences to convert building codes from natural language to computer-processable representations. POS tagging is about assigning the corresponding morphosyntactic category to each word in a sentence (Giménez and Marquez 2004). As an early step of the discussed automated code compliance checking system, POS tagging will cascade errors into later steps of the system (Dell'Orletta 2009) and jeopardize its final performance. An accurate POS tagging results of building codes is the foundation to support the high performance of the information extraction and transformation component and therefore the entire automated code compliance checking system.

Final published version is found in the ASCE Library here:
https://doi.org/10.1061/(ASCE)CP.1943-5487.0000917
Xue, X., and Zhang, J. (2020). “Building codes Part-of-Speech tagging performance improvement by error-driven transformational rules.” J. Comput. in Civ. Eng., 34(5), 04020035.

POS categories of words are classes of words that share common features (Brill 1992). In general, there are eight basic POS categories in English, namely, noun, pronoun, verb, adjective, adverb, preposition, conjunction and interjection (Butte College 2016). However, a decent representation of text for further NLP analysis needs more than just eight POS tags. For example, singular noun and plural noun are usually separated into two different categories. Among the commonly used tagset, Universal tagset has 12 tags (Petrov et al. 2011), Penn Treebank tagset has 36 tags (Marcus et al. 1993), and Brown tagset has 179 tags (Francis and Kucera 1979). The authors decided to use Penn Treebank tagset because of its good balance between information richness and conciseness.

There are multiple ways to get a textual corpus POS tagged. Human annotators can complete this task with their knowledge in English and understanding of the text. However, the high cost, low speed and human inconsistency make it rarely used in real-word applications. In contrast, POS tagging software, or POS taggers (will be called machine taggers hereafter) are usually used in NLP systems because of their fast tagging speed, low tagging cost, and free of human inconsistency. Machine taggers can tag a large amount of text in a short time without human interventions. The large amount of existing and upcoming building codes and frequent building codes updates require a machine POS tagging solution to support automated code compliance checking systems. POS taggers annotate texts according to rules or mathematical models. Correspondingly, there are two main types of machine POS taggers based on their corresponding annotation methodologies: rule-based POS taggers and machine learning POS taggers. These rules or models are either developed by humans or generated by algorithms.

Rule-based Part-of-Speech Tagger

Rule-based POS taggers decide POS tags of words based on a set of rules. Rules are designed to make POS tagging results of texts follow human-labeled results. These rules can be either hand-crafted by domain experts or generated by algorithms. Domain experts generate rules based on their understanding of English

Final published version is found in the ASCE Library here:
https://doi.org/10.1061/(ASCE)CP.1943-5487.0000917
Xue, X., and Zhang, J. (2020). "Building codes Part-of-Speech tagging performance improvement by error-driven transformational rules." J. Comput. in Civ. Eng., 34(5), 04020035.

grammar and the text being tagged. Rules can also be generated by algorithms. POS taggers with hand-crafted rules are rarely used nowadays. They usually are not intended for practical use but rather for educational purposes. For example, Bird et al. (2009) introduced a rule-based tagger with hand-crafted rules for educational purpose. However, this tagger has a low accuracy and only slightly outperformed a baseline tagger that tagged all words as "NNS" (plural nouns) (Bird et al. 2009). Development of rule-based POS taggers stopped because they, even with thousands of hand-crafted rules, fail to reach a comparable accuracy to that of machine learning taggers. For example, the TAGGIT system contains more than 3,000 hand-crafted rules and reached a 77% accuracy on Brown corpus (Greene and Rubin 1971), whereas the state-of-the-art machine taggers had an accuracy of 87.1% on Brown corpus which was much higher than the 77% accuracy achieved by TAGGIT (Li et al. 2012). However, rule-based POS tagger with algorithm-generated rules can achieve a higher accuracy than rule-based POS taggers with hand-crafted rules (Bird et al. 2009). For example, Brill (Brill 1992) developed the Brill tagger with algorithm-generated rules and claimed his tagger's performance "on par with stochastic taggers."

Machine Learning Part-of-Speech Tagger

Classification is one main task that machine learning was designed for. POS tagging is a type of classification task, i.e., classifying words into different POS categories according to its context and English grammar. Machine learning taggers are built by training machine learning models on corpus of English texts. Different machine learning models can be used such as support vector machines (SVM), decision tree, hidden Markov model (HMM), and neural network.

Methodology

The authors propose to use transformational rules to address errors in the tagging results of general POS taggers (i.e., machine taggers trained on general English texts) on building codes to increase their accuracy on POS tagging of building codes. Instead of training a new POS tagger from scratch, improving existing taggers can decrease the amount of annotated data needed, therefore save system development time and effort and potentially achieve higher POS tagging accuracy. The transformational rules are automatically generated by algorithms with no human intervention during the generation process execution.

In this paper, the authors define errors in POS tagging as nonconformities between the machine-assigned POS tag of a word and that word’s human-labeled tag. For example, machine taggers make a POS tagging error by tagging the word “can” in the phrase “a steel can,” which is a noun, as an “auxiliary verb.” Errors are further grouped into types. A type of error subsumes all appearances of a word in the textual data that have the same correct POS tag and are given the same incorrectly assigned POS tag by machine taggers. For example, for all occurrences of the word “can” as a noun, machine taggers may correctly tag them as a noun or incorrectly tag them as a modal verb or verb. For the occurrences that machine taggers incorrectly tagged the word “can” as a verb, it is one type of error. For the occurrences that machine taggers incorrectly tagged the word “can” as a modal verb, it is a different type of error. The proposed method focuses on decreasing the overall occurrence of errors, not specific types of errors. However, knowing possible types of errors is helpful to identify sources of errors. Furthermore, POS tagging errors in building codes textual data show a long-tail distribution. That is, a few types of errors happen many times and most types of errors only happen few times. In fact, for 1,758 types of 31,495 errors in the authors’ data of POS tagged building code where errors were defined to be the difference between machine tagging results and manually created gold standard, the top 100 types occurred 20,338 times, which accounted for 64.58% of all errors (Xue and Zhang 2020). The uneven distribution of errors implies that a small number of fixes may eliminate a large portion of errors.

Overview of the Method

The authors’ proposed method divides textual data into two parts, training dataset and testing dataset. The proposed method has two main components, rule generation component, and rule application component.

Final published version is found in the ASCE Library here:
https://doi.org/10.1061/(ASCE)CP.1943-5487.0000917
Xue, X., and Zhang, J. (2020). “Building codes Part-of-Speech tagging performance improvement by error-driven transformational rules.” J. Comput. in Civ. Eng., 34(5), 04020035.

The rule generation component uses rule templates to generate transformation rules. For example, “If the word B after the word A is tagged as X and the word A is tagged as Y, then change the tag of the word A to Z” is a rule template. All rules that are generated by the same template form a ruleset. This method allows users to input their customized templates to generate customized rulesets. The authors provided sample rule templates in the experiment section. The rule generation component generates rules from simple rulesets to complex rulesets, from uni-grams to n-grams, and from syntax to semantics. Before the development of each ruleset, the errors in the training set are collected and recorded. A process flowchart about error collection is shown in Figure 1. This process compares machine-generated tags of words and their corresponding human-labeled tags (from gold standard) in the training dataset, and records any word whose machine-generated tag is different from its human-labeled tag. If the machine-generated tag of the word “wood” is JJ (Adjective) and its human-labeled tag is NN (Noun), this method records the word “wood” is incorrectly tagged as JJ (Adjective) when it should be tagged as NN (Noun). This process is automatically and algorithmically performed by comparing the machine-assigned POS tag of a word and the human-labeled POS tag (from gold standard) of the same word, and recording any discrepancy between them for later steps of this method. After the error collection process, the rule generation process begins. The rule extraction component collects contextual information of errors in the training dataset and converts them to candidate transformational rules according to the template of that ruleset, and filters out unqualified rules. This is also automatically performed without human intervention. The proposed method will collect POS tags of words before and after the target word as the contextual information of the collected error. Before the extraction of the next ruleset, rules in the previous ruleset are applied to the training text. After the completion of rule development, all rulesets are applied to the testing dataset to evaluate the performance of the developed rules. The method also records remaining errors after each ruleset is applied to the testing dataset. The steps of this method are shown in Figure 2.

Final published version is found in the ASCE Library here:
https://doi.org/10.1061/(ASCE)CP.1943-5487.0000917
Xue, X., and Zhang, J. (2020). "Building codes Part-of-Speech tagging performance improvement by error-driven transformational rules." J. Comput. in Civ. Eng., 34(5), 04020035.

Description of Transformational Rules

The transformational rules fix POS tagging errors in the textual data. The POS tagging errors in the textual data are gathered by comparing machine tagging results of the textual data to the human-labeled gold standard. The rules store the word it matches and its contextual information, including semantic information (e.g., the word before the target word is "egress") and syntax information [e.g., the POS tag of the word before the target word is NN (noun)]. The proposed method utilizes two types of rulesets: n-gram rulesets that consider n-grams information of words and remaining error rulesets that consider remaining errors in the text. Rules in the N-grams rulesets also need to meet the rule acceptance criterion, which states that rules are not allowed to introduce any new errors in the training set.

N-grams Rulesets

N-gram rulesets are developed through the contextual information of errors in the training data. This paper does not differentiate bi-gram rules from n-gram rules. The authors treated them unanimously as n-gram rules. For example, "If machine tagger tags the word before 'pedestrian' as a noun and tags the token 'pedestrian' as an adjective, change POS tag of that prior word to adjective" is an n-gram rule. Each N-gram rule represents a context in which a word only has one possible correct POS tag. The context may include the word itself, the machine-assigned POS tag of the word, and machine-assigned POS tags of the word before and after a word.

Remaining Error Rulesets

After all n-gram rulesets are applied to the training data, a special ruleset is generated by fixing the n most common errors remaining in the training data. The choice of n is arbitrary. This special ruleset is special because the generation of rules in this set needs information from the entire training dataset whereas the generation of n-gram rulesets only need information from one sentence. The rule of thumb is that the user

Final published version is found in the ASCE Library here:
https://doi.org/10.1061/(ASCE)CP.1943-5487.0000917
Xue, X., and Zhang, J. (2020). "Building codes Part-of-Speech tagging performance improvement by error-driven transformational rules." J. Comput. in Civ. Eng., 34(5), 04020035.

can choose a larger n when there are more errors in the training dataset compared to when there are less errors. Different values of n can be tested to optimize the performance.

Rule Acceptance Test

To eliminate any potential negative effects of transformational rules on the downstream tasks of the automated code compliance checking system, any n-gram rules cannot introduce any new error to the textual data. The remaining error ruleset does not need to comply with this requirement. The rule acceptance test ensures an n-gram rule will not introduce any new error by making sure that the word only has one correct POS tag in the context described in the rule. If the word has more than one correct POS tag in the same context, all rules using that context will be dropped. Although it is mathematically true that a rule that fixes more errors than it introduces can increase the level of accuracy, the errors it introduces may undermine the performance of downstream tasks of the automated code compliance checking system in an unexpected way and drive the entire system further away from the 100% recall goal. Therefore, if a rule introduces new errors to the training set, even if it resolves more errors than it introduced, it failed to meet the rule acceptance criterion and will be left out from the ruleset. This strict requirement may limit the number of transformational rules generated, but it ensures a monotonous improvement of the quality of extracted rules and the rulesets' performance.

Rule Generation

The rule generation processes for each ruleset are similar. A general description of the rule generation procedure is shown in Figure 3. For each ruleset, the rule generation component collects contextual information of all errors and their corresponding human-labeled tags in the training dataset. In the second step, this component coverts collected information of each errors into candidate rules by deleting unnecessary contextual information. For example, if a rule only considers the POS tag of the word before

Final published version is found in the ASCE Library here:
https://doi.org/10.1061/(ASCE)CP.1943-5487.0000917
Xue, X., and Zhang, J. (2020). "Building codes Part-of-Speech tagging performance improvement by error-driven transformational rules." J. Comput. in Civ. Eng., 34(5), 04020035.

the target word, then only the target word, POS tag of the target word, and POS tag of the word before the target word will be kept and everything else in the target word's context will be deleted.

After that, all candidate rules need to undergo the rule acceptance test. This test clarifies the ambiguities in the textual data. One main challenge in POS tagging is that the same word may have different POS tags in different contexts. This test can ensure that the target word that a rule fixes only has one correct POS tag in the context described in the rule in the training dataset.

There are two scenarios that may occur in the generation of rules: all occurrences of a type of error have the same contextual information or have different contextual information. If all occurrences of a type of error share the same contextual information, this method will generate one candidate rule to fix all occurrences of this type of error. The candidate rule can pass the rule acceptance test and be included. If the same contextual information led to different rules, however, this indicates that the contextual information used was inappropriate. The rule acceptance test will prevent such candidate rules from being used, by grouping candidate rules with the same contextual information together and comparing them. Two scenarios may occur in this comparison: a word only has one correct POS tag in this captured context or has different possible POS tags in this captured context. There is no ambiguity in the first scenario. Replacing the machine generated tag with the correct POS tag will not introduce new errors. For example, the word "provided" only has one correct tag VBG in the training dataset when the POS tag of the word after it is DT (i.e., the word is "that") and it was incorrectly tagged by the machine taggers as VBN. In the second scenario, however, there is ambiguity. For example, the word "accessed" has two correct tags VBG and VBD in the training dataset, when it is incorrectly tagged by the machine taggers as VBN and the POS tag of the word after it is IN. Replacing the tag of "accessed" to either VBD or VBG entirely would introduce new errors, this indicates the captured context in this case (i.e., the POS tag of the word after it) is inappropriate and our method will not accept either rule in this scenario. Table 1 shows some example

Final published version is found in the ASCE Library here:
https://doi.org/10.1061/(ASCE)CP.1943-5487.0000917
Xue, X., and Zhang, J. (2020). "Building codes Part-of-Speech tagging performance improvement by error-driven transformational rules." J. Comput. in Civ. Eng., 34(5), 04020035.

sentences and candidate rules with regard to the above discussed scenarios. In this research, the authors adopted the widely used Penn Treebank POS tagset, which consists of 36 tags.

The decrease in the total number of errors only indicates a rule solved more errors than it introduced. It cannot ensure that a rule introduces no new errors. To check that, a detailed comparison between training datasets before and after applications of each rule is necessary after the generation of every single rule. However, the large number of possible rules and the large amount of calculation involved in a full-text comparison will extend the rule generation time to impractically long. The rule acceptance test used in this method can substantially save time necessary to generate rules by comparing candidate rules to find potential conflicts which will then be used to prevent rules in conflict from being added and therefore reduce the amount of rules to add.

Rule Application

In the rule application process (Figure 4), the rule application component will apply transformational rules to the textual data and fix POS tagging errors. For each rule, the rule application component will search through the entire text and look for words whose contextual information matches that rule's conditions. If a word's contextual information was found to match that rule's conditions, the rule application component will replace the machine-generated tag of that word with the predefined tag in the rule. After the generation of each ruleset, the developed ruleset is applied to the training dataset to prevent the rule application component from developing different rules that essentially fix the same error. After the generation of all rulesets, the rulesets are applied to the testing dataset as a whole.

Experiment

To test the performance of the proposed method on domain-specific data, the authors applied this method to the POS tagged building codes (PTBC) dataset (Xue and Zhang 2019). It contains 1,522 POS tagged

Final published version is found in the ASCE Library here:
https://doi.org/10.1061/(ASCE)CP.1943-5487.0000917
Xue, X., and Zhang, J. (2020). “Building codes Part-of-Speech tagging performance improvement by error-driven transformational rules.” J. Comput. in Civ. Eng., 34(5), 04020035.

sentences from Chapters 5 and 10 of the 2015 International Building Codes (IBC). For each tagged sentence, the dataset provides machine generated and human labeled POS tags of every token. In the formation of the PTBC dataset, the authors collected textual data by obtaining the Portable Document Format (PDF) version of 2015 IBC and manually extracted building code text from Chapters 5 and 10. A group of seven state-of-the-art machine taggers POS tagged the extracted texts. The seven selected POS taggers were: (1) the NLTK tagger (Loper and Bird 2002), (2) the spaCy tagger (Explosion 2015), (3) the Standford coreNLP tagger (Manning et al. 2014), (4) A Nearly-New Information Extraction System (ANNIE) tagger in the General Architecture for Text Engineering (GATE) tool (Cunningham 2002), (5) the Apache OpenNLP tagger (Kottmann et al. 2011), (6) the TreeTagger (Schmid et al. 2007), and (7) the RNNTagger (Schmid 2019). These taggers were chosen because they have high accuracy, are easy to use, and freely available. The most commonly chosen tag of each word in the extracted text by all the seven taggers formed the machine tagging results. The authors selected the Penn Treebank POS tagset because it was commonly used in various domains for NLP tasks and it is balanced between conciseness and informational richness. Five graduate students labeled textual data without access to others’ tagging results. All of them have proficiency in English and building domain knowledge to complete the tagging task, which ensures the quality of the textual data annotation. The mostly commonly chosen tag by them formed the gold standard of POS tagging of the textual data, with an inter-annotator agreement of 0.91.

The PTBC dataset was split into the training data, which contains 80% of the original dataset, and the testing data, which contains the remaining 20% of the original dataset. In the experiment, text is stored in lists of tuples (Figure 5). Each sentence is a list of tuples and each tuple in the list stores the word itself, human generated tag of the word, and machine generated tag of the word. In this experiment, the authors used possible combinations of contextual information of mistakenly tagged words in the textual data, to generate templates that rule generation component can use to extract rules. In total, fourteen templates were used in the experiment. They are listed in Table 2. The rule generation component extracted rules in the same order.

This method was also tested on the freely accessible portion of the Penn Treebank Corpus in the Natural Language Toolkit (NLTK) to further evaluate the applicability of the proposed method. The authors used the NLTK tagger to tag the text and collected the machine tagging results. Gold standard POS tags of the available text provided by the Penn Treebank Corpus served as the target of transformation. This comparative experiment was conducted in the same way as the previous experiment on PTBC data.

Experimental Results and Discussion

In total, on the PTBC data, 899 rules were generated in 14 rulesets. All extracted rules, when combined, fixed 3,003 out of 3,013 errors in the training dataset and 764 out of 924 errors in the testing dataset. They increased the tagging accuracy in the training dataset from 90.49% to 99.97% and that in the testing dataset from 89.13% to 98.12%. This 98.12% accuracy in testing dataset is comparable to the performance of the state-of-the-art POS taggers on general English corpus. The first three rulesets, which used contexts represented by: (1) the target word itself, (2) POS tag of the word two positions before the target word, and (3) POS tag of the word two positions after the target word, contained 825 rules (92.80% of all rules). In total, these first three rulesets fixed 2,961 errors (98.27% of errors) in the training dataset and 741 errors (80.19% of errors) in the testing dataset.

Accuracy of POS tagging both in the training dataset and in the testing dataset increased after application of the transformation rules. Before application of any transformational rules, the training dataset had an accuracy of 90.49% and the testing dataset had an accuracy of 89.13%. After all rulesets were applied, the training dataset achieved an accuracy of 99.97% and the testing dataset achieved an accuracy of 98.12%. The overall reduction of errors in the training set was 99.67% and that in the testing set was 82.68%. The most significant increase in accuracy happened after the application of the first and second rulesets. After the first ruleset was applied, accuracy in the training dataset increased from 90.49% to 97.60% and that in

Final published version is found in the ASCE Library here:
https://doi.org/10.1061/(ASCE)CP.1943-5487.0000917
Xue, X., and Zhang, J. (2020). “Building codes Part-of-Speech tagging performance improvement by error-driven transformational rules.” J. Comput. in Civ. Eng., 34(5), 04020035.

the testing dataset increased from 89.13% to 95.78%. After the second ruleset was applied, accuracy in the training dataset increased from 97.60% to 99.11% and that in the testing dataset increased from 95.78% to 97.29%. The number of errors and accuracy after application of each ruleset is provided in Table 3.

The authors recorded the number of errors each rule fixed to evaluate effectiveness of the generated rules. Ten rules that fixed the most errors fixed 52.34% errors in the training dataset and 38.10% errors in the testing dataset, respectively. This distribution confirms the authors’ prediction that a small group of rules can fix a large number of errors. Nine out of ten most frequently applied rules in the training dataset are uni-gram rules and that in the testing dataset is ten out of ten. This distribution shows that simple rules are more frequently applied than complex rules. It may not be necessary to generate over-complex rules in increasing POS tagging accuracy.

In the development of this method, the authors attempted to lemmatize word in text before the generation of transformational rules. The authors assumed that mapping multiple words to their common lemmatized form would improve the coverage of error cases. However, this generalization did not improve the performance and therefore the authors abandoned this technique. Word lemmatization actually caused a slight decrease in the number of extracted rules in all rulesets (i.e., 1.57% decrease on average). It is possible that mapping multiple forms of a word to one may have harmed the diversity of contextual information representation. With less fine-grained contextual information representation, it is harder to pinpoint contextual scenarios that only has one correct POS tag for a target word. The authors concluded that word lemmatization did not bring benefit to the proposed method.

This research also included a comparative study that applied the proposed method to improve NLTK POS tagger’s performance on Penn Treebank Corpus. This cross-comparison provides a useful benchmark for other researchers to compare this method’s performance on general English. In the processing of the Penn Treebank Corpus, the authors noticed that a none negligible amount of words in Penn Treebank Corpus, which do not belong to any Penn Treebank POS tagset, were tagged as ‘-none-’. Pre-processing Penn Treebank Corpus is a possible way to eliminate this ‘-none-’ tag. However, solving this issue is out of the scope of this paper. The authors decided to use the Penn Treebank Corpus and the NLTK tagging results in this method as is. The authors divided the Penn Treebank Corpus into a training dataset and a testing dataset with an 80/20 split. NLTK tagger tagged 89.28% of words in the training dataset correctly and 89.37% of words in the testing dataset correctly. The proposed method then increased the accuracy of NLTK tagger to 99.96% on the training dataset and to 96.47% in the testing dataset. This increase in accuracy indicates that the proposed method has the ability of improving the POS tagging accuracy of general English as well.

Discussion

Due to the specific type of texts covered in this research, the authors suggest that transformational rules should only be applied to texts that are in the target domain. A major potential risk is that transformational rules may introduce errors to the tagging results. This risk is eliminated by the rule acceptance test. This constraint can push the machine labeled result unidirectionally to the human labeled result.

Research interests of the authors require them to use the PTBC dataset, which is not used by other research currently. This method may overfit this particular dataset and lacks the ability to boost tagging accuracy of POS taggers, which are trained on general English, on general English. The authors conducted a comparative study to address this concern. They used this to boost the performance of Natural Language Toolkit (NLTK) tagger on the part of Penn Treebank Corpus that were readily available in NLTK (Loper and Bird 2002).

This method does not address unknown words. It requires a word to exist in the training set to generate transformational rules for it. This limitation, however, should not significantly influence the performance of the transformational rules, because generated rules are only to be applied to the text in the target domain (e.g., building codes), in which the rate of unknow words is expected to be low. The stringent format of the transformational rules in the proposed method, while effectively induced rules to improve POS tagging results, may introduce counter-intuitive tagging results. To alleviate that, future work may look into different representations of the fixes (e.g., tokens' roles) in addition to their original POS tags. In addition, the authors only tested their method on the commonly adopted Penn Tree bank tag set, how this method will perform when using other tag set will need to be investigated in the future work.

Contributions to the Body of Knowledge

This research presents a new way to get domain specific English texts POS tagged accurately when there is no POS tagger trained on that domain, by transformational rules. The proposed method can alleviate problems such as, (1) the lack of POS taggers that are trained on domain specific English texts, (2) the performance drop of general POS taggers on domain-specific texts, and (3) the high cost of developing a large domain specific corpus needed in training domain-specific POS taggers. This method provides a possible way for future researchers to get reliable POS tagged text in a selected domain without the need of a specialized POS tagger. The authors discovered that simple unigram and bigram rules resolved most errors. Word lemmatization did not bring observable benefit to this method. For future application of this method, development time could be saved by avoiding over-complicated rulesets and word lemmatization.

Secondly, this research proves that it is possible to boost the performance of POS taggers that are trained on general English texts on domain specific English texts with a small set of algorithmically generated rules. The authors use building codes as an example. These rules can increase the accuracy of POS taggers on building codes from 89.13% to 98.12% with 898 rules. This significant improvement is achieved by using a small set of labeled data. The fact that all rulesets transform machine-generated POS tags of words uni-directionally to their human-annotated tags proved the validity of the rule acceptance criterion. In addition, the increase in the accuracy in the testing dataset after the application of the last ruleset supports its exemption from the rule acceptance criterion.

Final published version is found in the ASCE Library here:
https://doi.org/10.1061/(ASCE)CP.1943-5487.0000917
Xue, X., and Zhang, J. (2020). "Building codes Part-of-Speech tagging performance improvement by error-driven transformational rules." J. Comput. in Civ. Eng., 34(5), 04020035.

Thirdly, the rules generated in this research can be used to increase the accuracy of POS tagging results on building codes. If interested researchers use one of the POS taggers tested, they can directly apply our developed rulesets to improve the POS tagging results/performance on building codes. The potential risk of introducing more errors were alleviated by the constraint applied when the rules were derived. This method does not need experts to generate new rules to be adapted to new domains, but it needs experts to annotate some training data as gold standard. Last but not least, this method is also applicable to general English. With a small amount of human-labeled data, it can boost the accuracy of POS taggers that are trained on general English, on general English.

Conclusions

This paper presented a new method to increase the accuracy of POS taggers, that were trained on general English texts, on building codes by using error-driven transformational rules. The authors developed an algorithm to generate these rules and tested the algorithm on PTBC data. The experiment shows this method can increase the POS tagging accuracy on building codes from 89.13% to 98.12%. A comparative test on NLTK and Penn Treebank Corpus shows that the proposed method can also increase the POS tagging accuracy on general English texts.

Data Availability

Some or all data, models, or code generated or used during the study are available in a repository or online in accordance with funder data retention policies.

1. Xue, X., Zhang, J. (2019). Part-of-Speech Tagged Building Codes (PTBC). Purdue University Research Repository. doi:10.4231/Y0ZQ-4946.*URL*: https://purr.purdue.edu/publications/3246/1)

Acknowledgement

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

Reference

Alghamdi, A., Sulaiman, M., Alghamdi, A., Alhosan, M., Mastali, M., and Zhang, J. (2017). “Building accessibility code compliance verification using game simulations in virtual reality.” *Computing in Civil Engineering 2017*, 262-270.

Bell, H., Bjorkhaug, L., and Hjelseth, E. (2009). “Standardized computable rules.” *National Office of Building Technology and Administration and Statsbygg, Oslo, Norway*.

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

Brill, E. (1992). “A simple rule-based part of speech tagger.” *Proc., Proceedings of the third conference on Applied natural language processing*, Association for Computational Linguistics, 152-155.

Butte College (2016). “The eight parts of speech.” <http://www.butte.edu/departments/cas/tipsheets/grammar/parts_of_speech.html>. (Sep 11st, 2019).

Ching, F. D., and Winkel, S. R. (2018). *Building Codes Illustrated: A Guide to Understanding the 2018 International Building Code*, John Wiley & Sons.

Choi, J., and Kim, I. (2017). *A Methodology of Building Code Checking System for Building Permission based on openBIM*.

Chollet, F. (2017). *Deep Learning with Python*, Manning Publications Co.

Final published version is found in the ASCE Library here:
https://doi.org/10.1061/(ASCE)CP.1943-5487.0000917
Xue, X., and Zhang, J. (2020). "Building codes Part-of-Speech tagging performance improvement by error-driven transformational rules." J. Comput. in Civ. Eng., 34(5), 04020035.

City of Chicago (2019). "Chicago construction codes." <https://www.chicago.gov/city/en/depts/bldgs/provdrs/bldg_code/svcs/chicago_buildingcodeonline.html>. (Sep 11st, 2019).

City of El Cajon Community Development Department (2019). "Fire permits." <https://www.cityofelcajon.us/your-government/departments/community-development/building-fire-safety-division/fire-permits>. (Sep 11st, 2019).

City of San Clemente (2019). "Ordinance No. 1668."San Clement, California.

City of Savannah (2019). "Building permit process flow chart." <https://www.savannahga.gov/DocumentCenter/View/3065/Building-Permit-Process-Flow-Chart?bidId=>. (Sep 11st, 2019).

Cunningham, H. (2002). "GATE, a general architecture for text engineering." *Computers and the Humanities*, 36(2), 223-254.

Delis, E. A., and Delis, A. (1995). "Automatic fire-code checking using expert-system technology." *Journal of computing in civil engineering*, 9(2), 141-156.

Dell'Orletta, F. (2009). "Ensemble system for part-of-Speech tagging." *Proceedings of EVALITA*, 9, 1-8.

Dimyadi, J., and Amor, R. (2013). "Automated building code compliance checking–where is it at." *Proceedings of CIB WBC*, 6.

Dimyadi, J., Clifton, C., Spearpoint, M., and Amor, R. (2016). "Computerizing regulatory knowledge for building engineering design." *Journal of Computing in Civil Engineering*, 30(5), C4016001.

Eastman, C., Lee, J.m., Jeong, Y.s., and Lee, J.k. (2009). "Automatic rule-based checking of building designs." *Automation in construction*, 18(8), 1011-1033.

Explosion, A. (2015). "Industrial-strength natural language processing." *Train. named entity recognizer*.

Fenves, S. J. (1966). "Tabular decision logic for structural design." *Journal of the Structural Division*, 92(6), 473-490.

Francis, W. N., and Kucera, H. (1979). “Brown Corpus Manual.”

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

Gero, J. S. (1990). “Design prototypes: a knowledge representation schema for design.” *AI magazine*, 11(4), 26-26.

Giménez, J., and Marquez, L. (2004). “Fast and accurate part-of-speech tagging: The SVM approach revisited.” *Recent Advances in Natural Language Processing III*, 153-162.

Greene, B. B., and Rubin, G. M. (1971). *Automatic Grammatical Tagging of English*, Department of Linguistics, Brown University.

İlal, S. M., and Günaydın, H. M. (2017). “Computer representation of building codes for automated compliance checking.” *Automation in Construction*, 82, 43-58.

Kottmann, J., Margulies, B., Ingersoll, G., Drost, I., Kosin, J., Baldridge, J., Goetz, T., Morton, T., Silva, W., and Autayeu, A. (2011). “Apache opennlp.” *Online (May 2011), www. opennlp. apache. org*.

Lee, Y.C., Ghannad, P., Shang, N., Eastman, C., and Barrett, S. (2018). “Graphical scripting approach integrated with speech recognition for bim-based rule checking.” *Construction Research Congress 2018*, 262-272.

Li, S., Cai, H., and Kamat, V.R. (2016). “Integrating natural language processing and spatial reasoning for utility compliance checking.” *Journal of Construction Engineering and Management*, 142(12), 04016074.

Li, S., Graça, J. V., and Taskar, B. (2012). “Wiki-ly supervised part-of-speech tagging.” *Proc., Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, Association for Computational Linguistics, 1389-1398.

Liao, S.H. (2005). “Expert system methodologies and applications—a decade review from 1995 to 2004.” *Expert systems with applications*, 28(1), 93-103.

Loper, E., and Bird, S. (2002). “NLTK: the natural language toolkit.” *arXiv preprint cs/0205028*.

Lopes, R., Bedwell, M., Stromberg, V., and Gottlieb, A. (2011). “Changing your hvac system? Don’t forget permits.” <https://ww2.energy.ca.gov/title24/2013standards/changeout/documents/2011-11-16_CSLB_News_Release_HVAC_Permits.pdf>. (Sep 11st, 2019).

Lopez, L., Elam, S., and Reed, K. (1989). “Software concept for checking engineering designs for conformance with codes and standards.” *Engineering with computers*, 5(2), 63-78.

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., and McClosky, D. (2014). “The Stanford CoreNLP natural language processing toolkit.” *Proc., Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, 55-60.

Marcus, M., Santorini, B., and Marcinkiewicz, M. A. (1993). “Building a large annotated corpus of English: The Penn Treebank.”

Nguyen, T.H., and Kim, J.L. (2011). “Building code compliance checking using BIM technology.” *Proc., Proceedings of the 2011 Winter Simulation Conference (WSC)*, IEEE, 3395-3400.

Petrov, S., Das, D., and McDonald, R. (2011). “A universal part-of-speech tagset.” *arXiv preprint arXiv:1104.2086*.

Preidel, C., and Borrmann, A. (2017). “Refinement of the visual code checking language for an automated checking of building information models regarding applicable regulations.” *Computing in Civil Engineering 2017*, 157-165.

Sabouni, A., and Al-Mourad, O. (1997). “Quantitative knowledge based approach for preliminary design of tall buildings.” *Artificial intelligence in Engineering*, 11(2), 143-154.

Sacks, R., Bloch, T., Katz, M., and Yosef, R. (2019). “Automating design review with artificial intelligence and bim: state of the art and research framework.” *Computing in Civil Engineering 2019*, 353-360.

Schmid, H. (2019). "Deep learning-based morphological taggers and lemmatizers for annotating historical texts." *Proc., Proceedings of the 3rd International Conference on Digital Access to Textual Cultural Heritage*, ACM, 133-137.

Schmid, H., Baroni, M., Zanchetta, E., and Stein, A. (2007). "The enriched treetagger system." *Proc., proceedings of the EVALITA 2007 workshop*.

Song, J., Kim, J., and Lee, J.K. (2018). "NLP and deep learning-based analysis of building regulations to support automated rule checking system." *Proc., ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, IAARC Publications, 1-7.

State of California, D. o. I. R. (2016). "Elevator permits." <https://www.dir.ca.gov/dosh/ElevatorPermits.html>. (Sep 11st, 2019).

Tan, X., Hammad, A., and Fazio, P. (2010). "Automated code compliance checking for building envelope design." *Journal of Computing in Civil Engineering*, 24(2), 203-211.

UK BIM Task Group (2016). "BIM level 2 frequently asked questions." <https://bim-level2.org/en/faqs/>. (Sep 11st, 2019).

Xu, X., and Cai, H. (2020). "Semantic approach to compliance checking of underground utilities." *Automation in Construction*, 109, 103006.

Xue, X., and Zhang, J. (2020). "Evaluation of eight part-of-speech taggers in tagging building codes: identifying the best performing tagger and common sources of errors." *Proc., The ASCE Construction Research Congress 2020*, ASCE, Reston, VA.

Xue, X., Zhang, J. (2019). "Part-of-speech tagged building codes (PTBC)." Purdue University Research Repository. doi:10.4231/Y0ZQ-4946.

Zhang, J., and El-Gohary, N. M. (2015). "Automated information transformation for automated regulatory compliance checking in construction." *Journal of Computing in Civil Engineering*, 29(4), B4015001.

Zhang, J., and El-Gohary, N. M. (2017). "Integrating semantic NLP and logic reasoning into a unified system for fully-automated code checking." *Automation in Construction*, 73, 45-57.

Tables

Table 1. Candidate Rules with and without Conflict

Scenario	Sentence	Candidate Rule
Without Conflict	The occupant load permitted in any building, or portion thereof, is permitted to be increased from that number established for the occupancies in Table 10, *provided (Manual tag: VBG; Machine tag: VBN) that (DT)* all other requirements of the code are met based on such modified number and the occupant load does not exceed one occupant per 7 square feet of occupiable floor space.	If the word that is one position after the word "provided" is tagged as DT and the word "provided" is tagged as VBN, then change the tag of the word "provided" to VBG.
	For auditoriums, theaters, concert or opera halls and similar assembly occupancies, the illumination at the walking surface is permitted to be reduced during performances by one of the following methods *provided (Manual tag: VBG; Machine tag: VBN) that (DT)* the required illumination is automatically restored upon activation of a premises' fire alarm system.	If the word that is one position after the word "provided" is tagged as DT and the word "provided" is tagged as VBN, then change the tag of the word "provided" to VBG.
With Conflict	Areas of refuge are not required for stairways *accessed (Manual tag: VBG; Machine tag: VBN) from (IN)* a refuge area in conjunction with a horizontal exit.	If the word that is one position after the word "accessed" is tagged as IN and the word "accessed" is tagged as VBN, then change the tag of the word "accessed" to VBG.
	Such open space shall be either on the same lot or dedicated for public use and shall be *accessed (Manual tag: VBD; Machine tag: VBN) from (IN)* a street or approved fire lane.	If the word that is one position after the word "accessed" is tagged as IN and the word "accessed" is tagged as VBN, then change the tag of the word "accessed" to VBD.

Final published version is found in the ASCE Library here:
https://doi.org/10.1061/(ASCE)CP.1943-5487.0000917
Xue, X., and Zhang, J. (2020). "Building codes Part-of-Speech tagging performance improvement by error-driven transformational rules." J. Comput. in Civ. Eng., 34(5), 04020035.

Table 2. Transformational Rulesets in the Experiment

Ruleset	Description
1	If the word A is tagged as X, then change the tag X to Y.
2	If the word that is one position before the word A is tagged as X and the word A is tagged as Y, then change the tag of the word A to Z.
3	If the word that is one position after the word A is tagged as X and the word A is tagged as Y, then change the tag of the word A to Z.
4	If the word that is one position before the word A is word B and the word A is tagged as X, then change the tag of the word A to Y.
5	If the word that is one position after the word A is word B and the word A is tagged as X, then change tag of the word A to Y.
6	If the word that is one position after the word A is tagged as X and the tag of the word that is two positions after word A is Y and the word A is tagged as Z, then change the tag of the word to W.
7	If the word that is one position after the word A is tagged as X and the tag of the word that is two positions after word A is Y and the word A is tagged as Z, then change the tag of the word A to W.
8	If the word one position before the word A is B, the word two positions before the word A is C, and the word A is tagged as X, then change the tag of word A to Y.
9	If the word one position after the word A is B, the word two positions after the word A is C, and the word A is tagged as X, then change the tag of the word A to Y.
10	If the tag of the word that is two positions after word A is X and the word is tagged as Y, then change the tag of the word A to Z.
11	If the tag of the word that is two positions before word A is X and the word is tagged as Y, then change the tag of the word A to Z.
12	If the word that is two positions after the word A is B and the word A is tagged as X, then change the tag of the word A to Y.
13	If the word that is two positions before the word A is B and the word A is tagged as X, then change the tag of the word A to Y.
14	Fix five most common errors remaining in the training set.

Table 3. POS Tagging Accuracy After Applying Each Ruleset

Ruleset	Training Dataset		Testing Dataset	
	Number of Errors	Accuracy	Number of Errors	Accuracy
1	759	97.60%	359	95.78%
2	282	99.11%	230	97.29%
3	142	99.55%	183	97.85%
4	91	99.71%	176	97.93%
5	80	99.75%	176	97.93%
6	36	99.89%	167	98.03%
7	36	99.89%	167	98.03%
8	29	99.91%	165	98.06%
9	29	99.91%	165	98.06%
10	29	99.91%	165	98.06%
11	29	99.91%	165	98.06%
12	29	99.91%	165	98.06%
13	29	99.91%	165	98.06%
14	10	99.97%	160	98.12%