Evaluation of Seven Part-of-Speech Taggers in Tagging Building Codes: Identifying the Best Performing Tagger and Common Sources of Errors

Xiaorui XUE[1] and Jiansong ZHANG[2]

[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; email: zhan3062@purdue.edu

ABSTRACT

As the number, size and complexity of building construction projects increase, code compliance checking becomes more challenging because of the time-consuming, costly, and error-prone nature of a manual checking process. A fully automated code compliance checking would be desirable in facilitating a more efficient, cost effective, and human error-proof code checking. Such automation requires automated information extraction from building designs and building codes, and automated information transformation to a format that allows automated reasoning. Natural Language Processing (NLP) is an important technology to support such automated processing of building codes, because building codes are represented in natural language texts. Part-of-speech (POS) tagging, as an important basis of NLP tasks, must have a high performance to ensure the quality of the automated processing of building codes in such a compliance checking system. However, no systematic testing of existing POS taggers on domain specific building codes data have been performed. To address this gap, the authors analyzed the performance of seven state-of-the-at POS taggers on tagging building codes and compared their results to a manually-labeled gold standard. The authors aim to: (1) find the best performing tagger in terms of accuracy, and (2) identify common sources of errors. In providing the POS tags, the authors used the Penn Treebank tagset, which is a widely used tagset with a proper balance between conciseness and information richness. An average accuracy of 88.80% was found on the testing data. The Standford coreNLP tagger outperformed the other taggers in the experiment. Common sources of errors were identified to be: (1) word ambiguity, (2) rare words, and (3) unique meaning of common English words in the construction context. The found result of machine taggers on building codes calls for performance improvement, such as error-fixing transformational rules and machine taggers that are trained on building codes.

INTRODUCTION

In recent years, many researches introduced automated building code compliance checking systems of various types (Eastman et al. 2009; Pauwels et al. 2011; Kasim 2013; Solihin and Eastman 2016; Zhang and El-Gohary 2016, 2017; Dimyadi and Armor 2017; Kim et al. 2018), to address the problems of traditional manual code compliance checking procedure: time-consuming, costly (City of Newark 2014), error-prone, and requiring specialized skills and years of experience (Town of Palm Beach 2019). For example, the city of Philadelphia charges a 350 dollars fee for preliminary plan review and it takes up to 20 business days to finish the review (City of Philadelphia 2019). A construction project may need to go through serval cycles of code reviews, which will make the total cost several times higher and the total duration much longer. Just in 2018,

there were more than 1.3 million new housing units authorized with building permits in the U.S. (United States Census Bureau 2019). The demand of plan review service stays high. The time-consuming, costly, and error-prone manual code compliance checking process already becomes a bottleneck of future productivity boosting in the architecture, engineering, and construction (AEC) industry and therefore calls for help by automation.

Automated code compliance checking systems can help address this demand and be extended to applications in infrastructure domain and beyond (Li et al. 2016). An automated code compliance checking system takes digital representations of building designs and that of building codes as inputs, and uses computer programs to find building designs' violations of building codes. The widespread use of Building Information Models (BIMs) provides high-quality and accessible digital representations of buildings. Many researchers adapted BIMs as digital representations of buildings (Nguyen and Kim 2011). However, the lack of appropriate digital representation and fully automated processing of building codes remains a major barrier on the way to an end-to-end fully automated code compliance checking system. Currently, many state-of-the-art automated code compliance checking systems still require manual extraction of regulatory information from building codes (Dimyadi and Amor 2013). Changing the publication of regulatory information to a way that computers can understand is a foreseeable solution (Hjelseth 2012). However, the current legislative establishment makes it difficult (if not impossible) to change how building codes are drafted in a foreseeable future. Even if such a shift occurs, it may take a long time period to update the existing building codes. Due to the difficulty of transforming the current way of drafting building codes and the large amount of existing codes, some researchers proposed automated information extraction systems, which can convert unstructured natural text building code requirements to structured data that computers can process (Salama and El-Gohary 2016; Zhang and El-Gohary 2016). These systems can achieve high processing accuracy with hand-crafted rules based on semantic NLP techniques. However, 100% recall of noncompliance detection is not achieved yet, which is a requirement to enable a practical application of such code compliance checking systems (Zhang and El-Gohary 2015).

To increase the recall of noncompliance detection in a semantic NLP-based code compliance checking system to 100%, every imperfect step in the system needs to be investigated for potential improvement. POS tagging, as a basis of NLP-based information extraction and transformation, demands a high accuracy to help achieve the desired performance of the entire code compliance checking system. In search of the most suitable POS tagger for this purpose, the authors tested the performance of seven state-of-the-art POS taggers using a manually labeled gold standard. Through this experiment and corresponding analysis, the authors aim to: (1) find the best performing POS tagger in terms of accuracy for processing building codes, and (2) identify common sources of POS tagging errors in tagging building codes.

BACKGROUND

In pursuit of the 100% recall goal of noncompliance detection in building code compliance checking, every imperfect step in an automated building code compliance checking system needs to be potentially improved. POS tagging, as a basis of the building code information extraction and transformation system, when reaching a high accuracy, can better support the entire system in achieving the 100% recall goal. Conversely, when it fails to meet this expectation, errors in this step will be cascaded into later steps (Dell'Orletta 2009). For all the POS taggers to be evaluated in this paper, they either provide online services that require users to upload raw text to a remote server and download tagged results to a local drive, or are able to tag all texts within a

short/reasonable timeframe (e.g., the Natural Language Toolkit (NLTK) tagger can tag a 30-word sentence in 0.1 seconds]. Because hardware requirement and tagging speed are not bottlenecks of tagger performance in this test, they are out of the scope of this paper. The authors focus on evaluating accuracy of the POS taggers.

EXPERIMENT

To evaluate the accuracy of different POS taggers, seven state-of-the-art POS taggers were selected: (1) the NLTK tagger (Loper and Bird 2002), (2) the spaCy tagger (Honnibal and Montani 2017), (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 (The University of Sheffield 2019), (5) the Apache OpenNLP tagger (The Apache Software Foundation 2019), (6) the TreeTagger (Schmid 2019), and (7) the RNNTagger (Schmid 2019). They were selected based on their: (1) reported high performance, (2) off-the-shelf availability, (3) free access, and (4) easiness to use. Five annotators, who were proficient in English, independently labeled the textual data used in the paper after receiving a short training. They provided a high-quality gold standard using an innovative annotating method that the authors developed. Their tagging results were published in the Purdue University Research Repository (PURR) for use in future researches. To compare the performance of different taggers and evaluate the reliability of human-labeled results, the authors used inter-annotator agreement as defined in Fleiss's Kappa (Fleiss 1971). Fleiss's Kappa is a statistical tool for measuring reliability of tagging results of more than two taggers. The widely used Cohen's Kappa is not directly applicable to this experiment because it can only compare tagging results of two taggers (Cohen 1960). This research used the Penn Treebank tagset, which was a commonly used tagset with a proper balance between conciseness and information richness. For the detailed meaning of all POS tags mentioned in this paper, the readers are referred to Penn Treebank corpus guideline (Marcus 1993). Some example POS tags are explained in Table 1.

Table 1: Example POS Tags

Tag	Meaning
DT	Determiner
IN	Preposition or subordinating conjunction
NN	Noun, singular or mass
NNP	Proper noun, singular
NNS	Noun, plural
RB	Adverb
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
WDT	Wh-determiner

Word Tokenization Disagreement Issue

To effectively compare the performances of different taggers, the most straightforward and intuitive way is to compare their tagging results word by word in a sentence. However, this simple

comparison sometimes cannot be easily achieved because different taggers may tokenize the same text in different ways. For example, some taggers tokenized 'second-floor' as a single word, whereas others tokenized it as three words: 'second', '-', and 'floor'. Some taggers tokenized 'cannot' as a single word, whereas others may tokenize it as two different words: 'can' and 'not'. A census showed there were less than 4 percent of sentences in the used textual data that had this issue. Because word tokenization is an integral step of POS tagging, inconsistency in word tokenization were treated as mistakes in POS tagging in the evaluation.

Textual Data

In this paper, the authors used *Chapter 5: General Building Heights and Areas*, and *Chapter 10: Means of Egress* of the International Building Code 2015 (IBC 2015) as the textual data to evaluate the performance of POS taggers. The authors manually extracted textual information in the PDF document of these two chapters and converted them to plain texts that taggers can process. In total, there were 1,522 sentences extracted from these two chapters.

An Innovative POS Tag Labeling Method for Gold Standard Development

POS labeling in a pure manual fashion is time-consuming, costly, and human error-prone (Manning 2011). In the creation of the Penn Treebank corpus (Marcus 1993), two main steps were taken to obtain POS-tagged text (i.e., labeled data) in a more efficient way: (1) a machine tagger POS tagged the text automatically, and (2) human annotators corrected the errors of machine tags manually. This method was more accurate and faster than letting human annotators tag the complete text manually, from scratch (Marcus 1993). Inspired by this method, the authors proposed and applied an innovative POS tag labeling method to reduce the workload of human annotators and improve the tagging accuracy. Annotators' main task in this method, in contrast to annotating from scratch, is resolving the differences among tagging results of different machine POS taggers. If all machine taggers assigned the same POS tag to a word, that tag is considered correct. This assumption is based on the fact that all taggers used had a claimed accuracy higher than 97%, thus the probability that they all made an error on the same word is a low probability event. Therefore, if all machine taggers tag one word identically, the annotators do not need to change it, unless they strongly disagree with it. If the taggers assigned different tags to a word, the annotators need to resolve this machine disagreement. To decrease the workload of human annotators, this method provides all tags assigned by machine taggers as options. However, annotators are free to choose any tag they deem correct.

The main advantage of this method is that it significantly decreases the workload of human annotators. Instead of selecting one tag from the entire tagset for all words in the text, the annotators just need to choose one tag from (usually) two or three tags, for a small portion of the text to be labeled. Annotators work on excel files that show all the words of each sentence in one column (Fig. 1), results of machine taggers (that are in agreement) in another column, and POS tag options (different tags assigned by different machine taggers) in the last column. Fig. 1 is an example of such a file. This annotation method can decrease the workload of annotators because they don't need to check the correctness of POS tags for each and every word. They only need to check the ones that were inconsistently tagged by machine taggers. In most cases, they just need to check a small portion of the textual data and select one tag from a few likely tags. A potential limitation of this method is that it is still possible (although a low probability event) that all taggers agree to a wrong tag or the correct tag is not selected by any of them, because machine taggers do not have a 100% accuracy. The authors addressed this issue by allowing annotators to change the

results of machine taggers even if all machine taggers tagged a word identically. The annotators are allowed to select tags that were not assigned by any of the machine taggers. The human-labeled gold standard uses the most commonly chosen POS tags of words in the textual data by human annotators.

The five annotators involved in this research were all Ph.D. students at Purdue University, proficient in English, and had sufficient AEC background knowledge to understand the textual data. To ensure the consistency of the tagging results, the annotators went through a short training on Penn Treebank POS tagset before the formal tagging process. The whole tagging phase was allotted 45 days, to provide annotators plenty of time to generate a high-quality tagging result. During the tagging process, the authors actively provided necessary answers and support to annotators. Their concerns and confusions about this task were resolved in a timely manner. After all tagging results were collected, the authors manually checked all of them to ensure that they were in a correct format, and corrected any formatting errors. In the end, all tagging results were stored in a single file for future research.

	Words	Tagging Consensus	POS Options
0	User		['NN', 'NNP']
1	note	NN	
2	:	:	
3	Code		['NN', 'NNP']
4	change		['NN', 'VB']
5	proposals	NNS	
6	to		['TO', 'IN']
7	sections	NNS	
8	preceded	VBN	
9	by	IN	
10	the	DT	
11	designation	NN	
12	will	MD	
13	be	VB	
14	considered	VBN	
15	by	IN	
16	the	DT	
17	International		['NNP', 'NP']
18	Fire		['NNP', 'NP']
19	Code		['NNP', 'NP']
20	Development		['NNP', 'NP']
21	Committee		['NNP', 'NP']
22	during	IN	
23	the	DT	
24	10	CD	
25	Code		['NNP', 'NP']
26	Development		['NNP', 'NP']
27	Cycle		['NNP', 'NP']
28	.	.	

Figure 1. Sample Annotator Work Space

RESULTS

Overall, the seven evaluated taggers reached an inter-annotator agreement of 0.93, indicating that they tagged the textual data with a high degree of consistency. At the same time, the inter-annotator agreement of human annotators was 0.91, showing that the human annotators also reached a high degree of consistency on their tagging task. Their tagging results formed the human-labeled gold standard used in this research where a majority vote mechanism was adopted.

The differences between machine tagging results and the human-tagged gold standard were considered tagging errors. The performance of all the seven machine taggers were shown in Fig.

1. It can be seen that the Standford coreNLP tagger had the least errors and therefore was identified as the most accurate tagger in tagging our building codes data, with an accuracy of 89.82%. The average accuracy of all evaluated POS taggers was 88.80%. The accuracy of all the seven taggers was shown in Fig. 2. In addition, when taking the most commonly assigned tag from the seven taggers, a better performance (90.20%) was achieved than any single POS tagger in the ensemble.

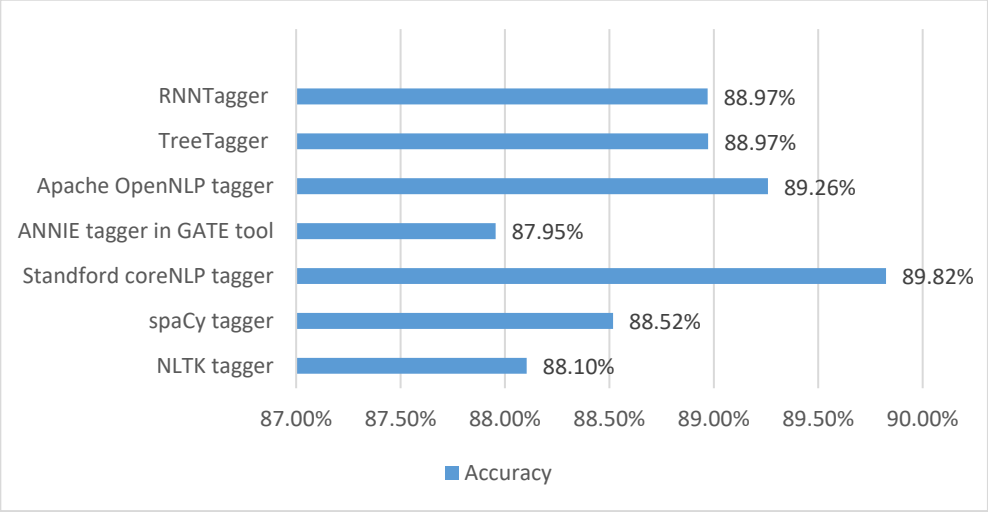

Figure 2: Accuracy of POS Taggers

Although all taggers reached a high performance (i.e., higher than 85%) on the textual data, they still assigned many incorrect tags. An error analysis was conducted to look into error patterns, and to seek for potential improvement for boosting the performance of the taggers to a higher level. After comparing results of machine taggers to the human-labeled gold standard, a tagging error report was generated that listed all errors made. In total, 1,758 different types of errors occurred 31,495 times. The most common type of tagging error was when a tagger incorrectly assigned the tag "VBN" (i.e., past participle verb) to the word 'required,' whose correct tag was "VBD" (i.e., past tense verb). In total, this error occurred 1,362 times. The top 100 most common errors happened 20,338 times, which took 64.58% of all the errors. This error distribution (Fig. 3) indicated that the taggers repetitively made a few types of errors. Therefore, a few countermeasures may correct a large portion of errors.

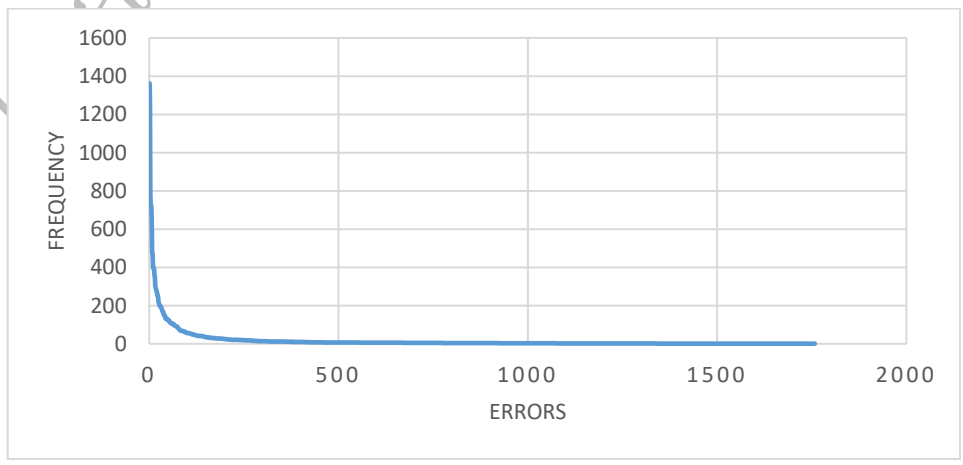

Figure 3: Distribution of Errors

After analyzing a random sample of 100 sentences in the tagging results of each tagger, the authors recognized three main categories of error causes: (1) word ambiguity - a word has multiple possible tags (e.g., most machine taggers tagged the word "required" VBN when it should be tagged VBD), (2) rare words (words that rarely appear in general English context but appear frequently in building codes, e.g., the word "egress"), and (3) unique meaning of common English words in the construction context, (e.g., the word "Fire" in the "International Fire Code Committee" refers to the name of a specific organization and should be NNP , but some machine tagger tagged it as NN). In the experimental results, 84% of errors were attributed to word ambiguity, 1% of errors were attribute to rare words, and 9% of errors were attributed to unique meanings of common English words in the construction context. The remaining 6% of errors could not be attributed to the categories above and were considered miscellaneous. For example, one machine tagger occasionally tagged double quotes ("") incorrectly as NN or NNP. When one part of the building code is referring to other parts of building codes, it may state "in accordance with Section A, B, and C". Machine taggers may tag "A", "B", and "C" as either NN, NNP or DT. Overall, word ambiguity contributes the most tagging errors. The most common type of error was when the correct tag of a word was VBD but it was tagged VBN incorrectly (Table 2). Furthermore, the taggers may be able to get most words tagged correctly, but a high performance at the word level (instead of at the sentence level) may still not be good enough to support a 100% recall goal in noncompliance detection for a code compliance checking system. For example, all evaluated taggers, when combined into an ensemble, could only tag 13.69% of sentences without error in the textual data used.

Table 2: Ten Most Common Tagging Errors

Word	Correct Tag	Wrong Tag	Frequency
required	VBD	VBN	1,362
Group	NN	NNP	1,253
permitted	VBG	VBN	785
that	DT	WDT	733
provided	VBG	VBN	722
as	RB	IN	705
feet	NN	NNS	581
Section	NN	NNP	474
located	VBD	VBN	470
occupant	JJ	NN	408

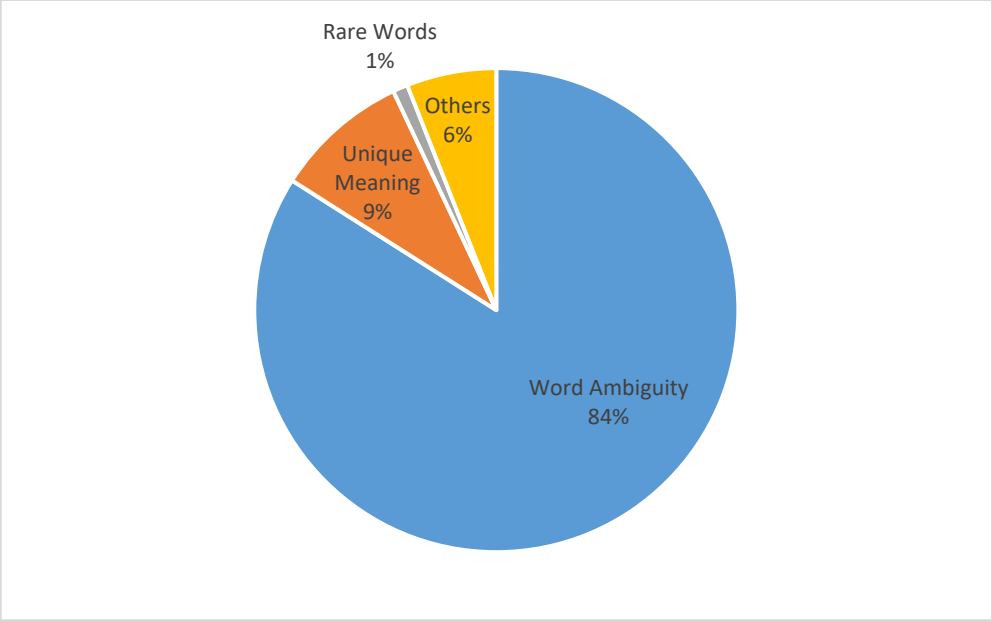

Figure 4: Common Sources of Errors

CONCLUSION

The state-of-the-art POS taggers can reach a decent accuracy on tagging sentences from building codes. Among seven POS taggers tested, the most accurate POS tagger was the Standford coreNLP tagger with a tested accuracy of 89.82%. The combined effort of a group of taggers achieved a better performance than that of any individual tagger in the group. Directly applying off-the-shelf POS taggers may not be sufficient to support the 100% recall goal in noncompliance detection in an NLP-based code compliance checking system. This performance gap calls for further researches and developments to fix errors of existing POS taggers or to create a new POS tagger that is trained on building codes. The authors also introduced a new, fast, and accurate POS data labeling method in this paper. It allows researchers to get reliable tagging results in a timely manner. Feedbacks from annotators have shown that most of them were able to complete the POS tagging task of the textual data within a reasonable timeframe. An error analysis in comparing machine tagging results with the human-labeled gold standard showed that machine taggers tend to confuse VBN, VBD, and VBG and different types of nouns in the building code text tested.

LIMITATIONS AND FUTURE WORK

This research only evaluated a few currently available POS taggers. It is possible that some other POS taggers may achieve a higher accuracy than those evaluated in this research. Also, the evaluated POS taggers did not show a desirable performance on building codes. In particular, their sentence-level accuracy was low. The authors plan to test more POS taggers and develop methods to improve POS taggers' performance on building codes.

ACKNOWLEDGEMENT

The author 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 author and do not necessarily reflect the views of the NSF.

REFERENCES

City of Philadelphia. (2019). "Preliminary plan review process – business services." *Business.phila.gov*, <https://business.phila.gov/preliminary-plan-review-process/>, (Apr. 19, 2019).

Cohen, J. (1960). "A coefficient of agreement for nominal scales". *Educational and Psychological Measurement*, 20(1), 37-46.

City of Newark. (2014). "Preliminary plan submittal." < http://www.newark.org/home/showdocument?id=68/>, (Jul. 31, 2019).

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

Dimyadi, J., and Amor, R. (2013). "Automated building code compliance checking - where is it at?" *Proc., 19th CIB World Building Congress, Brisbane 2013: Construction and Society*, Brisbane, Australia, 172-185.

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

Fleiss, J. (1971). "Measuring nominal scale agreement among many raters." *Psychological Bulletin*, 76(5), 378-382.

Kim, H., Lee, J., Shin, J., and Choi, J. (2018). "Visual language approach to representing KBimCode-based Korea building code sentences for automated rule checking." *Journal of Computational Design and Engineering, V(I),* 143-148.

Honnibal, M., and Montani, I. (2017). "spaCy 2: natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing." To appear.

Hjelseth, E. (2012). "Converting performance based regulations into computable rules in BIM based model checking software." *eWork and eBusiness in Architecture, Engineering and Construction*, Taylor & Francis Group, London, U.K., 461-469.

Kasim, T., Li, H., Rezgui, Y., and Beach, T. (2013). "Automated sustainability compliance checking process: proof of concept." *Proc., 13th International Conference on Construction Applications of Virtual Reality*, Teesside University, London, U.K., 10.

Nguyen, T., and Kim, J. (2011). "Building code compliance checking using BIM technology." *Proc., 2011 Winter Simulation Conference*, Winter Simulation Conference, Phoenix, AZ, 3400-3405.

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.

Loper, E., and Bird, S. (2002). "NLTK: The Natural Language Toolkit." *Proc., ETMTNLP '02 of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, Association for Computational Linguistics, Philadelphia, Pennsylvania, 63-70.

Marcus, M. (1993). "Building a large annotated corpus of English: the Penn Treebank." *Computational Linguistics - Special Issue on Using Large Corpora: II*, MIT Press, Cambridge, MA., 313-330.

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., and McClosky, D. (2014). "The Stanford coreNLP natural language processing toolkit." *Proc., 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Association for Computational Linguistics, Stroudsburg, PA, 55–60.

Manning, C. D. (2011). "Part-of-Speech tagging from 97% to 100%: is it time for some linguistics?" *Proc., CICLing'11 Proceedings of the 12th International Conference on Computational Linguistics and Intelligent Text Processing - Volume Part I,* Springer-Verlag, Berlin, Germany., 171–189.

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

Town of Palm Beach. (2019). "Combination plan reviewer." *Governmentjobs.com*, < https://www.governmentjobs.com/jobs/2514273-0/combination-plan-reviewer?utm_campaign=google_jobs_apply&utm_source=google_jobs_apply&utm_medium=organic/>, (Jul. 29, 2019).

Salama, D., and El-Gohary, N. (2016). "Semantic text classification for supporting automated compliance checking in construction." *Journal of Computing in Civil Engineering*, 30(1), 04014106.

Schmid, H. (2019). "TreeTagger." *Cis.uni-muenchen.de*, <http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/>, (Apr. 26, 2019).

Schmid, H. (2019). "RNNTagger." *Cis.uni-muenchen.de*, <http://www.cis.uni-muenchen.de/~schmid/tools/RNNTagger/>, (Apr. 26, 2019).

The Apache Software Foundation. (2019). "Apache OpenNLP." *Opennlp.apache.org*, <https://opennlp.apache.org/>, (Apr. 26, 2019).

The University of Sheffield. (2019). "English Part-of-Speech and morphology analyzer." *Cloud.gate.ac.uk*, <https://cloud.gate.ac.uk/shopfront/displayItem/pos-tagging-and-morphological-analysis>, (Apr. 26, 2019).

United States Census Bureau. (2019). *Building Permits Survey*. < https://www.census.gov/construction/bps/>, (Jul, 31, 2019).

Solihin, W., and Eastman, C. (2016). "A knowledge representation approach in BIM rule requirement analysis using the conceptual graph." *ITcon*, 21, 370-401.

Zhang, J., and El-Gohary, N. (2016). "Semantic NLP-based information extraction from construction regulatory documents for automated compliance checking." *Journal of Computing in Civil Engineering*, 30(2), 04015014.

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

Zhang, J., and El-Gohary, N. (2016). "A prototype system for fully automated code checking." *Proc., 16th International Conference on Computing in Civil and Building Engineering*, Osaka, Japan, 535-542.