Semantic Neural Network Ensemble for Automated Dependency Relation Extraction from Bridge Inspection Reports

Kaijian Liu, Ph.D., A.M.ASCE[1]; and Nora El-Gohary, Ph.D., A.M.ASCE[2]

Abstract: Bridge inspection reports are important sources of technically detailed data/information about bridge conditions and maintenance history, yet remain untapped for bridge deterioration prediction. To capitalize on these reports for improved bridge deterioration prediction, there is a need for dependency parsing methods, in order to extract dependency relations from the reports for linking the isolated words into concepts and representing the semantically low concepts in a semantically rich structured way. To address this need, this paper proposes a novel semantic neural network ensemble (NNE)–based dependency parsing methodology. It uses a similarity-based method to sample similarly distributed configurations into the same clusters, a set of constituent neural network (NN) classifiers to learn from both the syntactic and semantic text features of the similarly distributed and therefore more easily separable configurations, and a combiner classifier to capture the classification patterns of the NN classifiers to make final predictions on the transition types. The proposed dependency parsing methodology was evaluated in extracting dependency relations from bridge inspection reports for representing information—about bridge conditions and maintenance actions—in a semantically rich structured way. It achieved a precision, recall, and F-1 measure of 96.6%, 90.4%, and 93.3% with a margin of error of 3.8%, 4.4%, and 3.8% at the semantic information element level, and 88.2%, 81.5%, and 84.7% with a margin of error of 5.4%, 5.8%, and 5.4% at the semantic information set level, respectively. **DOI: 10.1061/(ASCE)CP.1943-5487.0000961.** *This work is made available under the terms of the Creative Commons Attribution 4.0 International license, https://creativecommons.org/licenses/by/4.0/.*

Author keywords: Bridges; Deterioration prediction; Natural language processing; Information extraction; Relation extraction; Dependency parsing; Neural network ensemble.

Introduction

According to the ASCE's Infrastructure Report Card, US bridges received a grade of C+ (mediocre), with 9.1% and 13.6% of the nation's 614,387 bridges being structurally deficient and functionally obsolete, respectively (ASCE 2017). A $123 billion investment in bridge maintenance is needed to eliminate the nation's deficient bridge backlog (ASCE 2017). Such critical conditions of bridges are not exclusive to the US but are faced by many countries across the world (Estes 2011). Bridge deterioration prediction is therefore of paramount importance for bridge management to predict deficiencies leading to the degradation of bridge conditions and to accordingly determine proactive bridge maintenance actions for improving the conditions of bridges in a cost-effective way.

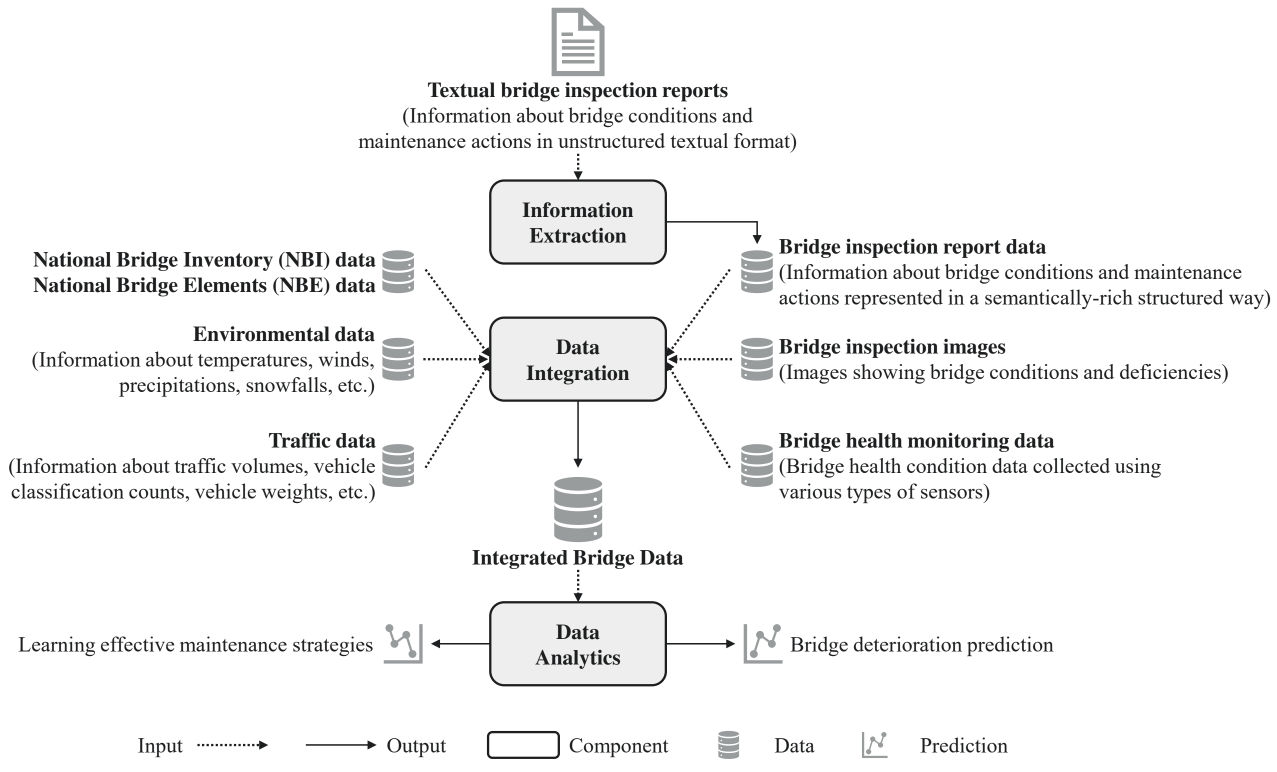
However, predicting bridge deterioration is challenging due to the large number of factors that affect the deterioration. Many mechanics-based prediction models (e.g., Nickless and Atadero 2018; Kameshwar and Padgett 2017; Ramanathan et al. 2015), which consider specific deterioration-related factors (e.g., corrosion and natural disaster), have been developed. With the increasing availability of bridge data that can capture multiple factors, there have been many demands for data-driven bridge deterioration prediction approaches (FHWA 2016; NASEM 2015). But the state-of-the-art data-driven prediction methods/models (e.g., Bu et al. 2014; Liu and Madanat 2014; Huang 2010; Mishalani and McCord 2006) mostly focus on learning from bridge characteristics and condition rating data [e.g., National Bridge Inventory (NBI) data] and can therefore only predict the future condition ratings of bridges. Condition rating data are certainly important, but not sufficient; they do not contain detailed information such as that found in bridge inspection reports. For example, bridge inspection reports include a large amount of detailed information about the deficiencies that caused bridge conditions to degrade, the maintenance actions that were used to mitigate these deficiencies, and their attributes (e.g., deficiency quantity and severity, maintenance material). This wealth of information, which is currently buried in the reports without being utilized, could offer unprecedented opportunities to data analytics for improving our abilities to better predict bridge deterioration, including the prediction of specific deficiencies and their severity levels and propagations in space and time, not only condition ratings.

To capitalize on this opportunity, the authors propose a novel smart bridge data analytics framework. The proposed framework includes three primary components, as per Fig. 1: (1) information extraction: information about bridge conditions and maintenance actions is extracted from unstructured textual bridge inspection reports and represented in a semantically rich structured way; (2) data integration: the extracted information is then integrated (linked and fused) with other bridge data (e.g., NBI data, environmental data, bridge inspection images, and bridge health monitoring data) to consider multiple sources of data/information and multiple factors affecting bridge deterioration and deficiency propagation; and

[1]Assistant Professor, Dept. of Civil, Environmental, and Ocean Engineering, Stevens Institute of Technology, 1 Castle Point Terrace, Hoboken, NJ 07030. Email: Kaijian.Liu@stevens.edu

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

Note. This manuscript was submitted on July 14, 2020; approved on October 26, 2020; published online on April 28, 2021. Discussion period open until September 28, 2021; separate discussions must be submitted for individual papers. This paper is part of the ***Journal of Computing in Civil Engineering***, © ASCE, ISSN 0887-3801.



Textual bridge inspection reports
(Information about bridge conditions and maintenance actions in unstructured textual format)
Information Extraction
Bridge inspection report data
(Information about bridge conditions and maintenance actions represented in a semantically-rich structured way)
National Bridge Inventory (NBI) data
National Bridge Elements (NBE) data
Environmental data
(Information about temperatures, winds, precipitations, snowfalls, etc.)
Data Integration
Bridge inspection images
(Images showing bridge conditions and deficiencies)
Traffic data
(Information about traffic volumes, vehicle classification counts, vehicle weights, etc.)
Bridge health monitoring data
(Bridge health condition data collected using various types of sensors)
Integrated Bridge Data
Learning effective maintenance strategies
Data Analytics
Bridge deterioration prediction
Input
Output
Component
Data
Prediction

Fig. 1. Proposed smart bridge data analytics framework.

(3) data analytics: the integrated data are analyzed using machine learning to predict future bridge condition states and specific bridge deficiencies and learn the most effective maintenance strategies. This paper focuses on the information extraction component. The data integration and analytics will be covered in the authors' future work.

Information extraction from bridge inspection reports is, however, a challenging task. The words in a sentence are isolated, needing to be linked to form meaningful concepts; and the subsequently linked concepts are semantically low, needing to be linked to the associated concepts to form a semantically rich structured representation of the information. There is therefore a need for dependency parsing methods to extract dependency relations from the reports, in order to represent the information in the reports in a semantically rich structured way that is ready for data analytics. For example, this sentence comes from a bridge inspection report (MnDOT 2006): "overlay has some minor spalls and patched areas around the finger joints, and 3,000 LF of transverse cracks." Dependency parsing is needed to extract the dependency relations to link the words *patched* and *areas* into the deficiency concept *patched_areas*; and then link *patched_areas* to the bridge element concept *overlay*, the categorical severity measure concept *minor*, and the categorical quantity measure concept *some* to represent the sentence in a semantically rich structured way: <*overlay*, *patched_areas*, *minor*, *some*>. Without dependency relations, it would be very challenging (if not impossible) to automatically infer from this unstructured sentence which bridge element (i.e., *overlay* or *finger_joints*) has which deficiency (i.e., *spall*, *patched_areas*, or *transverse_crack*) that is *minor* and *some*.

Existing dependency parsing methods are, however, not able to effectively extract dependency relations in such domain-specific and highly technical text—such as that in bridge inspection reports—for two main reasons. First, the current state-of-the-art dependency parsing methods (e.g., Dozat and Manning 2018; Strubell and McCallum 2017; Dozat and Manning 2017; Chen and Manning 2014) mostly rely on a single machine learning classifier to extract dependency relations. A single classifier is not sufficient in capturing the complex configuration distributions of the text in the bridge reports, because the reports are written by many different writers/inspectors from various agencies and are therefore highly variable in terms of text characteristics and patterns. An ensemble of classifiers usually performs better than a single classifier (Babbar and Schölkopf 2017; Zhang et al. 2011; Schiele 2002; Dietterich 2000), especially when dealing with high-dimensional data (Pes et al. 2017; Yu et al. 2017) and/or data with complex distributions such as imbalanced distributions (Haixiang et al. 2017; Bickel et al. 2007; Sun et al. 2006). Second, existing dependency parsing methods (e.g., Dozat and Manning 2018; Strubell and McCallum 2017; Dozat and Manning 2017; Chen and Manning 2014) typically only use syntactic features for supporting the extraction of dependency relations. But semantic text features are also very important for facilitating dependency parsing, because they provide semantics on word-to-word interactions that are critical when deciding on how sentences should be parsed. For example, based on the defined semantics that a categorical severity measure describes a bridge deficiency, the dependency relation between the concepts *minor* (as a modifier) and *patched_areas* (as a head) can be analyzed and extracted correctly.

To address this need, this paper proposes a novel semantic neural network ensemble (NNE)–based dependency parsing methodology. The objective of the methodology is to provide a way for effectively extracting dependency relations from domain-specific and highly technical text such as that in bridge inspection reports. The proposed methodology is built on the transition-based dependency parsing model, in which a sentence is represented by a set of configurations and the transition types of the configurations are sequentially classified for extracting the dependency relations in the sentence. The novelty of the NNE-based dependency parsing methodology lies in proposing and using a similarity-based method to sample similarly distributed configurations into the same clusters, a set of constituent neural network (NN) classifiers to learn from both the syntactic and semantic text features of the similarly distributed and therefore more easily separable configurations, and a combiner support vector machine (SVM) classifier to capture the classification patterns of the NN classifiers to make final predictions on the transition types.

Background

Transition-Based Dependency Parsing Model

Dependency parsing (DP) performs a grammatical structure analysis of a sentence to extract dependency relations between *head* words and their corresponding *modifier* words (Chen and Zhang 2015; Buchholz and Marsi 2006). Existing DP models can be categorized into graph-based and transition-based (McDonald and Nivre 2007). A graph-based model treats DP as a searching task in which subgraphs are factored, so that the model can search over the space of valid subgraphs to generate the most likely dependency graph (i.e., a set of dependency relations for a sentence) (Chen and Zhang 2015; Nivre and McDonald 2008). A transition-based model treats DP as a classification task, in which a set of configurations generated from an initial configuration is sequentially classified into transition types (indicating word-to-word dependency relations) for extracting dependency relations in a sentence (Chen and Manning 2014; Nivre and McDonald 2008). Transition-based DP models have gained considerable popularity because of their computational efficiency and accurate performance (Dyer et al. 2015; Weiss et al. 2015; Chen and Manning 2014; Choi and McCallum 2013).

The transition-based DP approach was introduced by Nivre (2003). As illustrated in Table 1, in the transition-based DP model, a configuration, $\boldsymbol{C} = (\boldsymbol{\sigma}, \boldsymbol{\beta}, \boldsymbol{A})$, is composed of a stack ($\boldsymbol{\sigma}$), a buffer ($\boldsymbol{\beta}$), and a set of dependency arcs/relations ($\boldsymbol{A}$). The stack, $\boldsymbol{\sigma} = [\sigma_i, \ldots \sigma_2, \sigma_1]$, where $i \geq 0$, is a data structure that stores partially parsed words of an input sentence. The buffer, $\boldsymbol{\beta} = [\beta_1, \beta_2 \ldots, \beta_j]$, where $j \geq 0$, is a data structure that stores the words of the sentence that need to be parsed. The set $\boldsymbol{A}$ is a data structure that stores word pairs that have been parsed with dependency relations. The initial configuration of the input sentence is defined as $\boldsymbol{C} = (\boldsymbol{\sigma} = [Root], \boldsymbol{\beta} = [\beta_1, \beta_2 \ldots, \beta_n], \boldsymbol{A} = \emptyset)$, where $Root$ is a dummy node at the highest level of a dependency graph and $\beta_1, \beta_2 \ldots, \beta_n$ correspond to the words of the sentence (where n is the length of the sentence). The terminal configuration of the sentence is defined as $\boldsymbol{C} = (\boldsymbol{\sigma} = [Root], \boldsymbol{\beta} = \emptyset, \boldsymbol{A})$, where $\boldsymbol{A}$ contains the parsed dependency relations of the sentence. From the initial configuration, the transition-based model predicts a transition type for the current configuration and generates the next configuration based on the current configuration and the predicted transition type. This process repeats until some terminal configuration has been reached, which indicates that the sentence has been completely parsed. Three transition types are defined in the transition-based DP model, including:

- *Shift*: moving β_1 from the buffer $\boldsymbol{\beta}$ to the stack $\boldsymbol{\sigma}$, if $|\boldsymbol{\beta}| \geq 1$.
- *Left-arc*: adding an arc between σ_1 and σ_2, where σ_1 is a head word and σ_2 is a modifier word, and removing σ_2 from the stack $\boldsymbol{\sigma}$, if $|\boldsymbol{\sigma}| \geq 2$.
- *Right-arc*: adding an arc between σ_1 and σ_2, where σ_2 is a head word and σ_1 is a modifier word, and removing σ_1 from the stack $\boldsymbol{\sigma}$, if $|\boldsymbol{\sigma}| \geq 2$.

Machine Learning–Based Dependency Parsing Methods

Early DP research efforts (e.g., Oflazer 2003; Elworthy 2000; Tapanainen and Järvinen 1997) have focused on developing rule-based DP methods. Rule-based DP methods utilize manually developed parsing rules to extract dependency relations. More recently, machine learning–based DP methods have been proposed for automatically classifying configurations into transition types for dependency relation extraction. Some of these efforts have focused on developing probabilistic models (e.g., Wang and Harper 2004; Collins 2003; Samuelsson 2000; Eisner 1996), while others have proposed discriminative approaches with support vector machines (e.g., Kudo and Matsumoto 2002; Yamada and Matsumoto 2003), beam search–based perceptron (e.g., Zhang and Nivre 2011;

Table 1. Example of a transition-based dependency parsing model

Transition	Stack	Buffer	Arc (head, modifier)
	[Root]	[The bottom chord connection of truss has severe crevice corrosion]	
S	[Root The]	[bottom chord connection of truss has severe crevice corrosion]	
S	[Root The bottom]	[chord connection of truss has severe crevice corrosion]	
S	[Root The bottom chord]	[connection of truss has severe crevice corrosion]	
S	[Root The bottom chord connection]	[of truss has severe crevice corrosion]	
L	[Root The bottom connection]	[of truss has severe crevice corrosion]	(connection, chord)
L	[Root The connection]	[of truss has severe crevice corrosion]	(connection, bottom)
L	[Root connection]	[of truss has severe crevice corrosion]	(connection, The)
S	[Root connection of]	[truss has severe crevice corrosion]	
S	[Root connection of truss]	[has severe crevice corrosion]	
L	[Root connection truss]	[has severe crevice corrosion]	(truss, of)
R	[Root connection]	[has severe crevice corrosion]	(connection, truss)
S	[Root connection has]	[severe crevice corrosion]	
L	[Root has]	[severe crevice corrosion]	(has, connection)
S	[Root has severe]	[crevice corrosion]	
S	[Root has severe crevice]	[corrosion]	
S	[Root has severe crevice corrosion]	[ ]	
L	[Root has severe corrosion]	[ ]	(corrosion, crevice)
L	[Root has corrosion]	[ ]	(corrosion, severe)
R	[Root has]	[ ]	(has, corrosion)
R	[Root]	[ ]	(Root, has)

Note: S = shift; L = left arch; and R = right arch.

Zhang and Clark 2008), dynamic programming–based perceptron (e.g., Huang and Sagae 2010), or neural networks (e.g., Mayberry and Miikkulainen 2005; Henderson 2004).

In recent years, there have been an increasing number of research efforts focusing on NN-based DP methods (e.g., Dozat and Manning 2018; Strubell and McCallum 2017; Nguyen et al. 2017; Dozat and Manning 2017; Hashimoto et al. 2017; Kuncoro et al. 2017; Kiperwasser and Goldberg 2016; Cheng et al. 2016; Yazdani and Henderson 2015; Zhou et al. 2015; Alberti et al. 2015; Weiss et al. 2015; Dyer et al. 2015; Chen and Manning 2014). Neural networks have gained popularity in the area of DP for two main reasons. First, as opposed to conventional machine learning–based DP methods (which rely heavily on handcrafted indicator features), NN-based DP methods can automatically learn the most useful feature conjunctions and high-order features, which helps avoid feature sparsity and incompleteness issues (Pei et al. 2015; Chen and Manning 2014). Second, DP can benefit from neural networks by learning from NN-based distributed feature representations. Distributed feature representations (also known as word embedding) transform text features [e.g., words and part-of-speech (POS) tags] into real-valued, continuous, and dense vectors, and embed semantically similar features near each other in the vector space (Mikolov et al. 2013). Such representations result in a compact dense feature space, which leads to more efficient, compact, and accurate classifier learning (Chen and Manning 2014). Recent efforts (e.g., Guo et al. 2015; Bansal et al. 2014; Chen and Manning 2014) have demonstrated that, compared to learning from traditional one-hot feature representations, learning from NN-based distributed feature representations can improve DP performance.

The work by Chen and Manning (2014) is one of the first efforts that incorporated neural networks and deep learning into a transition-based DP model (Dozat and Manning 2017). They developed a simple, yet relatively accurate and computationally efficient, three-layer feedforward NN architecture for supporting general domain DP applications. Many NN-based DP methods that used more complex NN architectures have since been developed to further improve the parsing accuracy, such as the recurrent neural network (Kuncoro et al. 2017), the long short-term memory (LSTM) (Kiperwasser and Goldberg 2016), and the bi-LSTM with deep biaffine attention (Dozat and Manning 2017). Compared to the three-layer feedforward NN architecture, these complex architectures were able to marginally improve the parsing accuracy, but at the expense of computational efficiency (see Dozat and Manning 2017; Chen and Manning 2014).

Ensemble Machine Learning Methods

Ensemble machine learning is a learning paradigm that utilizes multiple classifiers to obtain improved performance (reduced variability and increased generalization) that cannot be obtained by any of the constituent classifiers alone (Sun 2013; Zhang and Ma 2012). The most well-established and prominent ensemble learning algorithms include bagging, boosting, stacked generalization, and mixture of experts (Xu et al. 2013; Zhang and Ma 2012). Bagging trains each of the multiple classifiers with a certain percent of instances that are randomly drawn with replacement from the entire training set (Breiman 1996). Boosting sequentially trains a set of classifiers, each of which focuses on learning from the instances that were misclassified by its preceding classifier (Schapire 1990). Adaptive boosting, also referred to as AdaBoost, is a widely known boosting algorithm (Raghavan et al. 2019; Bui et al. 2018). It sequentially trains a set of classifiers, during which the initial classifier is trained with instances sampled based on a uniform distribution and each of the subsequent classifiers is trained with instances sampled according to a weighted distribution, in which the weight is updated based on the distribution and training errors of its preceding classifier (Freund and Schapire 1995). Stacked generalization first trains a set of tier-1 classifiers with training instances sampled using cross-validation partitioning, and then trains a tier-2 combiner classifier using the outputs of the tier-1 classifiers as input (Wolpert 1992). The combiner classifier aims to learn the misclassification and/or classification patterns to correct the misclassifications generated by the tier-1 classifiers. A mixture of experts trains a set of classifiers (experts) and a gating network that allocates an individual instance to one or several classifiers (Jacobs et al. 1991). The outputs of the selected classifier(s) are then combined through a linear rule to yield a final classification decision for the instance.

State of the Art and Knowledge Gaps

There is a body of research efforts that have focused on developing machine learning–based dependency parsing (DP) models—using different learning techniques and various feature representations—for extracting dependency relations in text. Despite the importance of these efforts, they cannot effectively extract dependency relations in domain-specific and highly technical text such as that in bridge inspection reports. Three primary knowledge gaps are identified in this regard.

First, from a machine learning–based DP perspective, there is a lack of studies in ensemble learning–based DP methods. The majority of such methods (e.g., Dozat and Manning 2018; Strubell and McCallum 2017; Nguyen et al. 2017; Dozat and Manning 2017; Hashimoto et al. 2017; Kuncoro et al. 2017; Kiperwasser and Goldberg 2016; Cheng et al. 2016; Dyer et al. 2015; Chen and Manning 2014; Zhang and Nivre 2011; Zhang and Clark 2008; Yamada and Matsumoto 2003) have focused on learning a single classifier to parse text for extracting dependency relations. Although a single classifier trained with advanced learning techniques (e.g., SVM and NN) could perform well on nonlinearly separable instances/configurations, it is not sufficient to separate those with even more complex distributions (Haixiang et al. 2017; Bickel et al. 2007; Sun et al. 2006) such as the configurations of the text in the bridge reports (especially given that the reports have highly varying levels of text characteristics and patterns). There are several efforts (e.g., Hall et al. 2010; Attardi and Dell'Orletta 2009; Sagae and Lavie 2006; Nivre and McDonald 2008) that have proposed to integrate DP models at the parser level. For example, Nivre and McDonald (2008) proposed to integrate a graph-based parser and a transition-based parser by letting one parser generate features for the other. Such methods are more cotraining-based rather than ensemble learning–based. To the authors' best knowledge, there is no ensemble learning–based DP method that utilizes a set of constituent classifiers to collectively capture the complex distributions of all the configurations for improved dependency relation extraction performance.

Second, from an ensemble learning perspective, there is a lack of studies in sampling training instances/configurations in a way that each constituent classifier is trained only with similarly distributed and therefore more easily separable configurations. As discussed in the "Ensemble Machine Learning Methods" section, existing ensemble learning techniques sample instances based on simple, presumed distributions, such as the uniform distribution or weighted uniform distribution. Sampling configurations in this way cannot capture the configuration distribution characteristics of the text in bridge inspection reports, which makes it hard to generate meaningful configuration clusters and could therefore make the trained constituent classifiers limited in collectively and sufficiently capturing the underlying distributions of all the configurations.

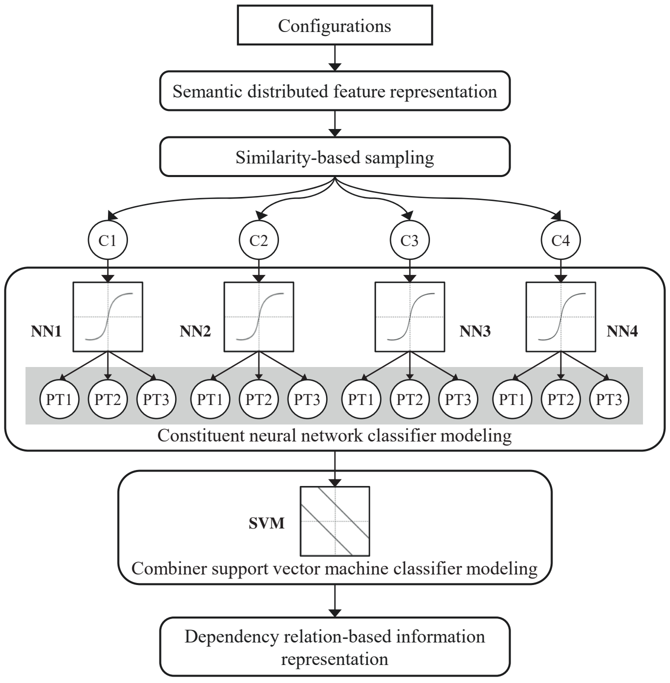
Third, from a feature representation perspective, there is a lack of studies that utilized semantic text features for facilitating DP. Existing DP methods (e.g., Dozat and Manning 2018; Guo et al. 2015; Chen and Manning 2014; Bansal et al. 2014) have relied on using distributed representations of syntactic features (e.g., words and POS tags). Although distributed feature representations could reveal the semantic meanings of the features to some extent, they provide limited semantics about word-to-word interactions that are important to consider when deciding on how sentences should be parsed. Such interactions can be better captured by the semantic features. For example, *maintenance material* and *maintenance action* are the semantic features for the words *concrete* and *patching*, respectively. Based on the defined semantics—a maintenance material concept semantically describes a maintenance action concept—the dependency relation between *concrete*, as a modifier word, and *patching*, as a head word, could be correctly parsed and extracted.

Proposed Semantic Neural Network Ensemble–Based Dependency Parsing Methodology

To address the aforementioned knowledge gaps, a semantic neural network ensemble (NNE)–based dependency parsing (DP) methodology is proposed. The proposed methodology is composed of five primary components, as per Fig. 2: semantic distributed feature representation, similarity-based sampling, constituent NN classifier modeling, combiner SVM classifier modeling, and dependency relation–based information representation. The proposed methodology is novel in three primary ways. First, it proposes a new feature representation for the configurations, which includes both syntactic (words and POS tags) and semantic (the semantic classes of words) text features. The semantic features aim to capture the semantics about the word-to-word interactions for facilitating the extraction of dependency relations. Second, it proposes and utilizes a new similarity-based sampling method to capture the distribution characteristics of the configurations and sample the similarly distributed configurations into the same clusters. Compared to existing sampling methods used in ensemble learning (see the "Ensemble Machine Learning Methods" section), the proposed method can better capture how the configurations distribute. It generates more meaningful configuration clusters that contain the densely and sparsely distributed as well as the correctly and incorrectly densely distributed configurations, which facilitates the classifier ensembling and the NNE-based DP. Third, the proposed DP methodology takes an ensemble learning–based approach. It uses a set of constituent NN classifiers to collectively capture the complex distributions of all the configurations and utilizes a combiner SVM classifier to capture the classification and/or misclassification patterns of the NN classifiers for making final predictions on the transition types. Each of the constituent classifiers only learns from similarly distributed and therefore more easily separable configurations. The ensemble of the classifiers can better capture the complex distributions, which are challenging for a single classifier to capture (Haixiang et al. 2017; Bickel et al. 2007; Sun et al. 2006).

Semantic Distributed Feature Representation

A new semantic distributed feature representation is proposed to represent the configurations. As shown in Fig. 3, it is a multilevel


Configurations
Semantic distributed feature representation
Similarity-based sampling
C1
C2
C3
C4
NN1
NN2
NN3
NN4
PT1
PT2
PT3
Constituent neural network classifier modeling
SVM
Combiner support vector machine classifier modeling
Dependency relation-based information representation

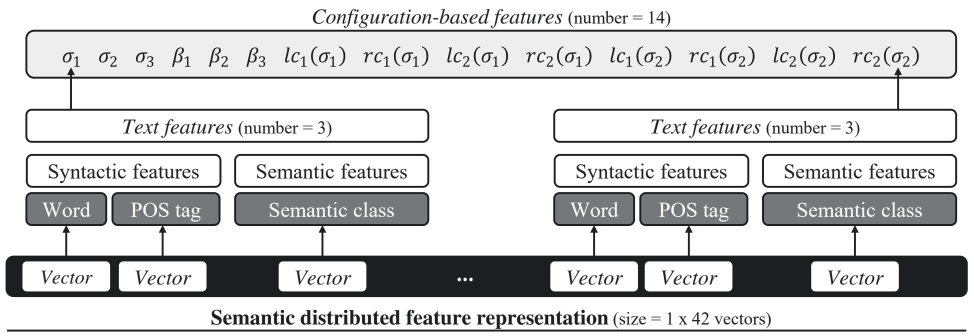
C1 = majority cluster; C2 = minority cluster; C3 = correct-majority cluster; C4 = incorrect-majority cluster; NN = neural network; PT = the probability of a transition type ("Shift", "Right-arc", or "Left-arc"); SVM = support vector machine.

Fig. 2. Proposed semantic neural network ensemble–based dependency parsing methodology.

representation. First, the configurations are represented by the configuration-based features. These features are defined according to the positions of the elements (words of a sentence) in a configuration (Zhang and Nivre 2011). The configuration-based features include 14 features: (1) the top three elements of the stack: σ_1, σ_2, σ_3; (2) the top three elements of the buffer: β_1, β_2, β_3; (3) the first and second leftmost/rightmost children of the first element in the stack: $lc_1(\sigma_1)$, $rc_1(\sigma_1)$, $lc_2(\sigma_1)$, $rc_2(\sigma_1)$; and (4) the first and second leftmost/rightmost children of the second element in the stack: $lc_1(\sigma_2)$, $rc_1(\sigma_2)$, $lc_2(\sigma_2)$, $rc_2(\sigma_2)$. The top three words are used to capture the contextual information of the word pair to be parsed (i.e., the first word of the stack and the first word of the buffer) (Zhang and Clark 2008; Zhang and Nivre 2011).

Second, each of the configuration-based features is represented by syntactic and semantic text features. The syntactic features include: (1) words: the original lexical forms of the words; and (2) POS tags: the lexical classes of the words, which are defined based on the syntactic structures of the sentences. The semantic features are the semantic classes of the words. In this research, to capture the semantics about the word-to-word interactions in the text and the information that needs to be extracted and represented, the following semantic classes were defined by the authors based on an analysis of a sample of bridge inspection reports: bridge element (ET), deficiency (DY), deficiency cause (DC), numerical measure (NM), numerical measure unit (NU), categorical quantity measure (QM), categorical severity measure (SM), maintenance action (MA), maintenance material (MM), and date (DT).

Third, the text features are further represented using distributed feature representations. For example, instead of using *noun* as the POS tag for the word *crack*, the NN-based distributed feature representation utilizes a vector with a user-defined vector size to


Configuration-based features (number = 14)
σ_1 σ_2 σ_3 β_1 β_2 β_3 $lc_1(\sigma_1)$ $rc_1(\sigma_1)$ $lc_2(\sigma_1)$ $rc_2(\sigma_1)$ $lc_1(\sigma_2)$ $rc_1(\sigma_2)$ $lc_2(\sigma_2)$ $rc_2(\sigma_2)$
Text features (number = 3)
Syntactic features
Semantic features
Word
POS tag
Semantic class
Vector
Semantic distributed feature representation (size = 1 x 42 vectors)

Note: The symbols for configuration-based features are defined in the "Semantic Distributed Feature Representation" Section. POS = part-of-speech. In the experiments of this research, a vector of size 50 was used, and the size of the representation is 2100 (i.e., 50 x 3 x 14 = 2100).

Fig. 3. Proposed semantic distributed feature representation.

represent it numerically. Therefore, using the proposed feature representation, a configuration is represented by a numeric vector of size 2,100: 14 configuration-based features, 3 text features for each of the configuration-based features, and a vector of size 50 for each of the text features (i.e., $14 \times 3 \times 50 = 2{,}100$).

Similarity-Based Sampling

The configuration distributions of the text exhibit the following characteristics: (1) a majority of the configurations are distributed in a dense area; (2) a minority of them are distributed in a sparse area; and (3) in the dense area, the configurations of a gold standard transition (GST) type (*shift*, *left-arc*, or *right-arc*) overlap with the configurations of the other GST types. Because of the overlapping, some of the configurations distribute relatively far away from the center of their corresponding GST type and relatively close to one of the other centers, where a center is the arithmetic mean of all the configurations that belong to the same GST type. To capture these characteristics in a way that each constituent NN classifier will be trained only with the similarly distributed and therefore more easily separable configurations, the authors propose to sample the configurations into one or two of the following four configuration clusters:

- $C1$: This is a majority cluster, which contains the densely distributed configurations that belong to all the GST types and are distributed close to one of the centers, where $C1 = C3 \cup C4$.
- $C2$: This a minority cluster, which contains the sparsely distributed configurations that belong to all the GST types and are distributed far away from all the centers.
- $C3$: This a correct-majority cluster, which contains the densely distributed configurations that belong to all the GST types and are distributed close to the center of their corresponding *correct* GST type, where $C3 \in C1$.
- $C4$: This an incorrect-majority cluster, which contains the densely distributed configurations that belong to all the GST types and are distributed close to the center of another *incorrect* GST type, where $C4 \in C1$.

The $C1$ and $C2$ clusters aim to differentiate the densely distributed configurations from the sparsely distributed ones. The $C3$ and $C4$ clusters aim to differentiate the densely distributed configurations in $C1$—differentiating those that distribute close to the center of their correct GST type from those that distribute close to the center of an incorrect GST type. Although $C1 = C3 \cup C4$, sampling into $C1$ besides into $C3$ or $C4$ makes a one-to-one correspondence for $C1$ and $C2$ as well as for $C3$ and $C4$, which helps explicitly capture the aforementioned distribution characteristics.

To sample all the configurations into the aforementioned clusters, the authors propose a similarity-based sampling method. Similarities between the configurations and their centers are indicative of the distribution characteristics. For example, if a configuration is similar to (is close to) the center of a GST type, it is sampled into the $C1$ cluster. However, the similarity measured in one feature space is insufficient to capture the complex distribution characteristics, because different degrees of similarities (measured distances for indicating *being close* or *being far away*) emerge when some other features are used for the measurement (Harispe et al. 2015). To deal with this issue, the authors propose to measure the similarities in seven different feature spaces, and utilize the similarities measured in these spaces collectively as a criterion to sample the configurations into the defined clusters. These feature spaces are defined in Table 2.

The proposed similarity-based sampling method is summarized as follows. First, the centers of the configurations of the three GST types are computed in each feature space. A center is computed by calculating the arithmetic mean of all the configurations (in the proposed feature representation) that belong to the same GST type and are in the same feature space. As a result, a total of 21 centers (for three GST types and seven feature spaces) are generated. Second, a configuration gets associated with a similarity-based transition (ST) type in each feature space. In a space, the ST type of a configuration is the GST type of the center that is most similar to the configuration compared to the other two centers, where the similarity degree is computed by the cosine similarity measure. The ST and GST types of a configuration could be same or different, because the configuration could be closer to the center of a correct or an incorrect GST type. As a result, a configuration gets associated with a total of seven ST types, one per feature space. Third, the configurations are sampled into the aforementioned clusters based on

Table 2. Defined feature spaces

Feature space	Features[a]
1	Words
2	POS tags
3	Semantic classes
4	Words + POS tags
5	Words + semantic classes
6	POS tags + semantic classes
7	Words + POS tags + semantic classes

Note: POS = part-of-speech.
[a]All the features are in their distributed representations.

Eq. (1), where $CMST$ is the count of the majority similarity-based transition, MST is the type of the majority similarity-based transition, and GST is the gold standard transition type.

For a configuration, Eq. (1) works as follows. First, if the count of the majority ST of the configuration is greater than or equal to the *natural threshold* (i.e., four out of seven), the configuration is sampled into the $C1$ cluster; otherwise, it is sampled into the $C2$ cluster. This is because in the former case the majority of the STs have reached a consensus, which indicates that the configuration can be confidently associated close to one of the centers; while in the latter case no consensus has been made, which indicates that the configuration cannot be confidently associated close to any of the centers. Second, if a $C1$ configuration happens to have an MST type that is same as its GST type, it is sampled into the $C3$ cluster as well; otherwise, it is sampled into the $C4$ cluster. This is because in the former case the majority of the STs are indicating a correct transition type (the GST type of the configuration), while in the latter case no correct transition type can be decided based on the MST type

$$\text{Cluster} = \begin{cases} C1 & \text{if } CMST \geq 4; \\ C2 & \text{if } CMST < 4; \\ C3 & \text{if } CMST \geq 4 \text{ and } MST = GST; \\ C4 & \text{if } CMST \geq 4 \text{ and } MST \neq GST. \end{cases} \quad (1)$$

Constituent Neural Network Classifier Modeling

An NN architecture was modeled and developed for training a set of constituent NN classifiers. It is a feedforward neural network that contains an input layer, a hidden layer, and an output layer. This NN architecture was chosen for two reasons. First, it can automatically learn the most useful feature conjunctions and high-order features, which helps avoid feature sparsity and incompleteness issues (Pei et al. 2015; Chen and Manning 2014). Second, it does not use a complex neural network topology, which helps balance classification accuracy and computational efficiency (Chen and Manning 2014).

The input layer takes the semantic distributed feature representation of a configuration as input. A unit of the input layer takes a value from the representation. Based on the size of the semantic feature representation vectors (see the "Semantic Distributed Feature Representation" section), the input layer has a size of 2,100. The hidden layer contains a set of hidden units, each of which is fully connected to the input layer. A hidden unit takes a value mapped from the input layer. The mapping is conducted by an activation function. For instance, using the logistic sigmoid function as an example, a hidden unit has an input value of h_i that is computed by Eq. (2), where $W_1 \in R^{|X| \times |H|}$ is a weight matrix, $B_1 \in R^{|H|}$ is a bias vector, $|X|$ is the size of the input layer, and $|H|$ is the size of the hidden layer. In this research, a hidden layer size of 200 (Chen and Manning 2014) was used, and the logistic sigmoid function was selected and used based on the experimental results (see the "Hyperparameter Value Selection" section). The output layer is a softmax layer added upon the hidden layer and is used to model the multiclass probabilities of a configuration being classified into the transition types. The probabilities are computed by Eq. (3), where t_j is the jth transition type, $W_2 \in R^{|3| \times |H|}$ is a weight matrix, and $B_2 \in R^{|3|}$ is a bias vector. Based on the number of transition types in the transition-based DP model, the output layer has a size of 3. For a data set $D = \{(c^k, t^k)\}_{k=1}^{K}$, where c^k is the kth configuration and t^k is its corresponding GST type, the training process of the NN architecture aims to minimize the L_2-regularized cross-entropy loss (maximizing the probabilities of the training configurations being classified into their GST types). The loss function is defined in Eq. (4), where $\theta = \{W_1, B_1, W_2, B_2\}$ and λ is a regularization parameter

$$h_i = \frac{1}{1 + \exp(-W_{1i}X - B_{1i})}, \quad \text{for } i = 1, \ldots, |H| \quad (2)$$

$$P_{t_j} = \frac{\exp(W_{2j}h + B_{2j})}{\sum_{i=1}^{3} \exp(W_{2i}h + B_{2i})}, \quad \text{for } j = 1, 2, 3 \quad (3)$$

$$L(\theta) = -\sum_k \log P_{t^k} + \frac{\lambda}{2} \|\theta\|^2 \quad (4)$$

Combiner Support Vector Machine Classifier Modeling

A combiner SVM classifier was modeled and developed. It aims to capture the misclassification and/or classification patterns of all the constituent NN classifiers, and to make final configuration classification decisions for extracting dependency relations from the text. As shown in Fig. 2, the combiner SVM classifier takes the outputs of the four constituent NN classifiers (three probabilities per constituent classifier; see the "Constituent Neural Network Classifier Modeling" section) as input. Therefore, the input of the combiner classifier is a probability vector of size 12. Training a classifier in such case is a straightforward learning process, because the input contains less features and simple patterns, and the resulting learning process does not involve extensive feature conjunctions and mappings. SVM has shown high performance in such learning tasks (e.g., Shibuya et al. 2015; Priya and Aruna 2012), and was therefore chosen for training the combiner classifier.

Dependency Relation–Based Information Representation

A dependency relation–based information representation method is proposed. It aims to decode the extracted word-to-word dependency relations, in order to link the isolated words into semantic information elements (SIEs) and to represent the unstructured and semantically low SIEs into semantically rich structured semantic information sets (SISs). In this research, an SIE is a concept that describes bridge conditions and maintenance actions, which could be a bridge element (ET), deficiency (DY), deficiency cause (DC), numerical measure (NM), numerical measure unit (NU), categorical quantity measure (QM), categorical severity measure (SM), maintenance action (MA), maintenance material (MM), or date (DT). An SIS is a semantic information structure that consists of SIEs. The SIEs in an SIS must follow an SIE-to-SIE dependency relation type. For example, as illustrated in Fig. 4, the SIEs must follow one of the three SIE-to-SIE dependency relation types: (1) the ET-DY dependency relation with the semantics: a *bridge element* is affected by a *deficiency* that is inspected at a *date*, that is caused by a *deficiency cause* and is maintained by a *maintenance action* using a *maintenance material*, and that has a *numerical measure* with a *numerical measure unit*, a *categorical severity measure*, and a *categorical quantity measure*; (2) the ET-DC dependency relation with the semantics: a *bridge element* has a *deficiency cause* that is inspected at a *date*; and (3) the ET-MA dependency relation with the semantics: a *bridge element* is maintained by a *maintenance action* using a *maintenance material* at a *date*.

The proposed dependency relation–based information representation method, as illustrated in Fig. 5, contains three main steps. First, a sentence is represented with a sequence of words, semantic classes, word numbers, and head word numbers. The word and

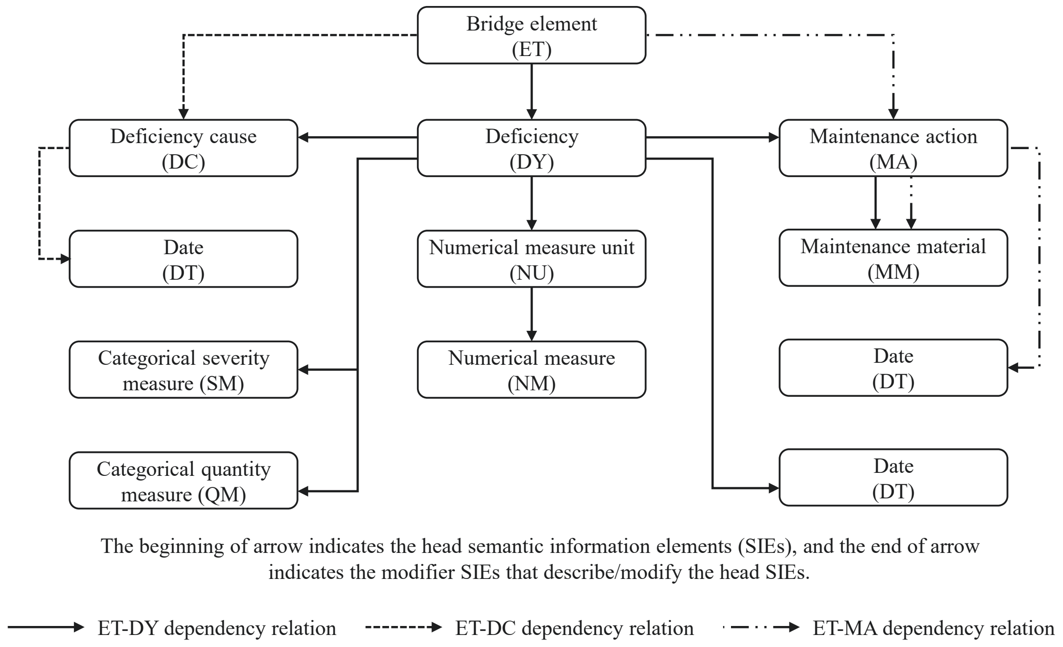
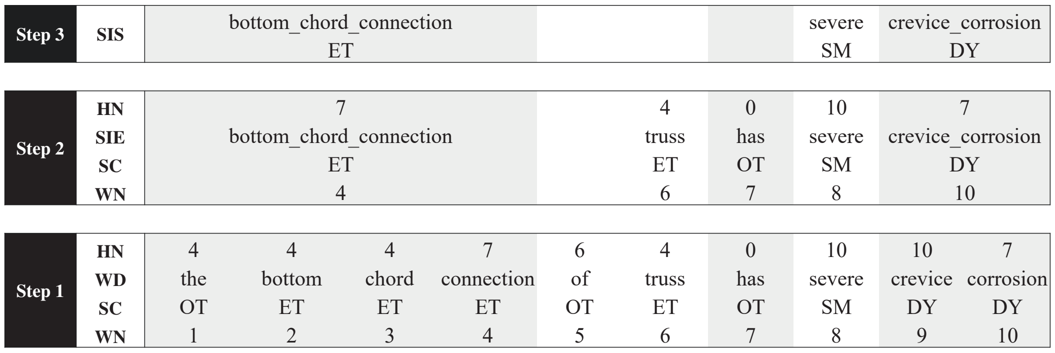

Bridge element (ET)
Deficiency cause (DC)
Deficiency (DY)
Maintenance action (MA)
Date (DT)
Numerical measure unit (NU)
Maintenance material (MM)
Categorical severity measure (SM)
Numerical measure (NM)
Date (DT)
Categorical quantity measure (QM)
Date (DT)
The beginning of arrow indicates the head semantic information elements (SIEs), and the end of arrow indicates the modifier SIEs that describe/modify the head SIEs.
ET-DY dependency relation
ET-DC dependency relation
ET-MA dependency relation

Fig. 4. SIE-to-SIE dependency relations defined in a semantic information set (SIS).



Step 3										
SIS	bottom_chord_connection ET							severe SM	crevice_corrosion DY	

Step 2										
HN	7					4	0	10	7	
SIE	bottom_chord_connection					truss	has	severe	crevice_corrosion	
SC	ET					ET	OT	SM	DY	
WN	4					6	7	8	10	

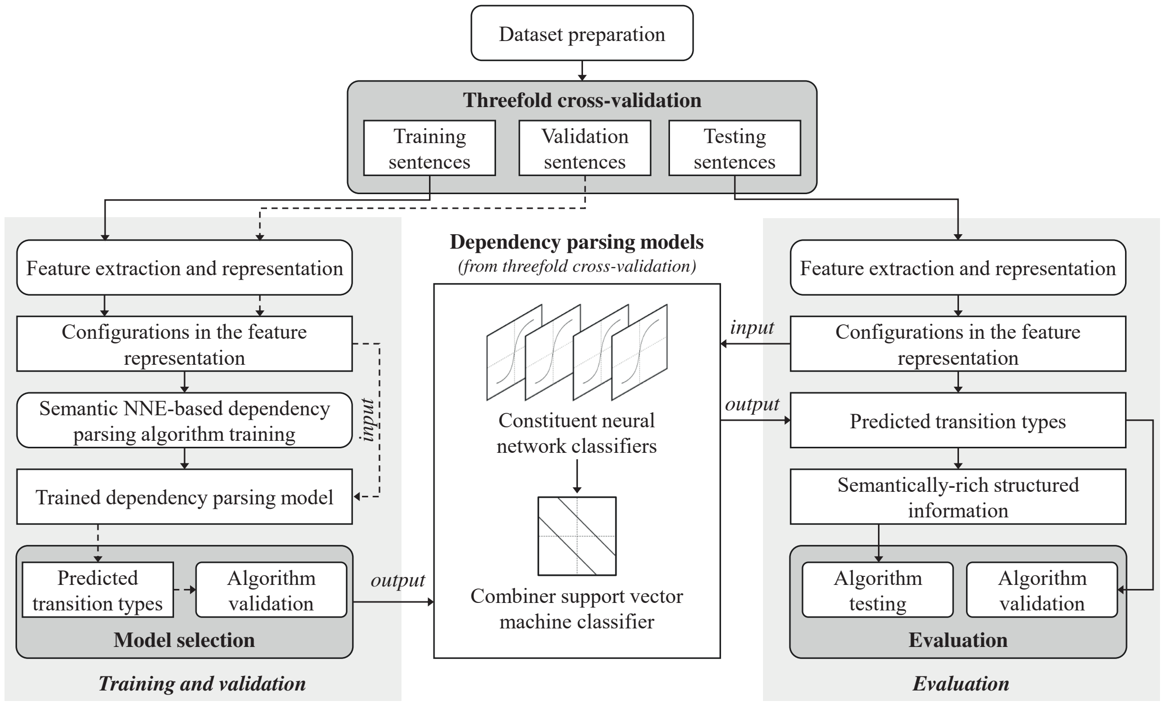
Step 1										
HN	4	4	4	7	6	4	0	10	10	7
WD	the	bottom	chord	connection	of	truss	has	severe	crevice	corrosion
SC	OT	ET	ET	ET	OT	ET	OT	SM	DY	DY
WN	1	2	3	4	5	6	7	8	9	10

SIS = semantic information set; SIE = semantic information element; HN = head word number; WD = word; SC = semantic class; WN = word number; OT = other; ET = bridge element; SM = categorical severity measure; DY = deficiency.

Fig. 5. Example to illustrate proposed dependency relation–based information representation method.

head word numbers indicate the extracted word-to-word dependency relations. For example, in Fig. 5, the head word of *chord* is at position 4, which is *connection*. Second, starting from the left-hand side of a sentence, the modifiers with the consecutively same head word numbers and the head word are combined to form SIEs, and their corresponding semantic classes are combined to form the semantic classes of the SIEs. For example, in Fig. 5, although the word *truss* also has a head word number of 4, it was not combined with *the bottom chord* because the consecutively same head word numbers of 4 are broken by the numbers of 7 and 6. In this step, only the word and the head word numbers of the original head words are maintained. For example, in Fig. 5, the modifier words *bottom* and *chord* were combined with the head word *connection* to form the SIE (i.e., *bottom chord connection*) and the semantic class (i.e., *ET*) with the word and head word numbers of 4 and 7, respectively. The semantic classes of the SIEs are needed in order to associate the right SIEs into the right positions of an SIS, and to break down the SIEs that contain concepts with different semantic classes. For example, in Fig. 5, the phrase *severe crevice corrosion* was further broken down into *severe* and *crevice corrosion* SIEs based on their semantic classes (*SM* and *DY*, respectively). Third, the extracted SIE-to-SIE dependency relations are checked to assess whether they follow the SIE-to-SIE dependency relations as defined in Fig. 4, so that only valid SIEs are added to an SIS. For example, in Fig. 5, the SIE pair *bottom chord connection* and *truss* was excluded only because there are no dependency relations defined between the two ET SIEs. ET-ET dependency relation can be defined and used, if extracting information about bridge elements and their constituent parts (e.g., a bottom chord connection is a part of a truss) is desired in one's application.

Two special cases that include conjunction and negation are also considered in the proposed information representation method. First, if one SIE is dependent on the other SIE and they are concatenated by a conjunction, they inherit the dependency relations of each other. For example, in the following sentence from (DRJTBC 2016), *abutment* and *deck* have a dependency relation and are concatenated by a conjunction (i.e., *and*): "Leaching at corner of north abutment and bottom of deck." In this case, *deck* inherits the dependency relations of the *abutment* and gets associated with *leaching* as well. Second, if an SIE is concatenated to another SIE



Dataset preparation
Threefold cross-validation
Training sentences
Validation sentences
Testing sentences
Feature extraction and representation
Configurations in the feature representation
Semantic NNE-based dependency parsing algorithm training
Trained dependency parsing model
input
Predicted transition types
Algorithm validation
Model selection
Training and validation
output
Dependency parsing models
(from threefold cross-validation)
Constituent neural network classifiers
Combiner support vector machine classifier
input
output
Feature extraction and representation
Configurations in the feature representation
Predicted transition types
Semantically-rich structured information
Algorithm testing
Algorithm validation
Evaluation
Evaluation

NNE = neural network ensemble. Algorithm testing was conducted only for the proposed algorithm (with the selected hyperparameters and feature representation).

Fig. 6. Overview of implementation of proposed semantic neural network ensemble–based dependency parsing methodology.

by a negation, both SIEs (and their associated SIEs) are excluded from an SIS. For example, in the following sentence from (Caltrans 2012), *connections* was concatenated with *distress* by a negation (i.e., *did not*): "The connections did not appear to be in distress." In this case, these SIEs are excluded from an SIS. To capture conjunctions and negations, two gazetteer lists were developed and used. The conjunction gazetteer list includes words/phrases such as *and*, *as well as*, *along with*, and *together with*. The negation gazetteer list includes words/phrases such as *no*, *not*, *doesn't*, and *isn't*.

Implementation of the Proposed Methodology

The proposed semantic NNE-based dependency parsing (DP) methodology was implemented in extracting dependency relations from bridge inspection reports for linking the isolated words that describe bridge conditions and maintenance actions into SIEs and SISs. The implementation included four primary steps: data set preparation, feature extraction and representation, semantic NNE-based DP algorithm training, and evaluation. An overview of the implementation methodology is presented in Fig. 6.

Data Set Preparation

Data set preparation included data set creation, text preprocessing, and human annotation. A data set, which contains a total of 1,000 sentences that were randomly selected from 10 bridge inspection reports, was created. As given in Table 3, the selected reports are from different states, from different reporting years, and for different bridge structure types. The sentences were randomly selected to avoid introducing bias. To further ensure that the sample (the selected sentences) is representative of the population (all the sentences from the 10 reports), the distributions of the sentence lengths were compared. As shown in Fig. 7, the two distributions are quite similar. The p-value for the comparison of the distributions is 0.4892 (calculated from the Welch's unequal variances t-test, assuming normal distributions of the sentence lengths), which shows that there is no significant difference between the two. These results

Table 3. List of bridge inspection reports

No.	Reported bridge	Structure type	State	Year	Ref.
1	Natchaug River Chaplin Bridge	Concrete arch bridge	Connecticut	2009	ConnDOT (2009)
2	Sherman Minton Bridge	Double-deck through arch bridge	Indiana	2007	INDOT (2007)
3	Hale Boggs Memorial Bridge	Cable-stayed bridge	Louisiana	2008	LaDOTD (2008)
4	Heron Truss Bridge	Steel deck truss bridge	Montana	2011	MDT (2011)
5	Portsmouth Memorial Bridge	Vertical-lift bridge	New Hampshire	2009	NHDOT (2009)
6	Wellwood Avenue Bridge	Concrete arch bridge	New York	2015	NYSDOT (2015)
7	Union Street Railroad Bridge	Vertical-lift, Pratt through truss bridge	Oregon	2005	ODOT (2005)
8	South Park Bridge	Scherzer rolling lift double-leaf bascule bridge	Washington	2009	WSDOT (2009)
9	Lower Trenton Bridge	Through truss bridge	New Jersey	2015	DRJTBC (2016)
10	Capitola Crossing Deck Truss	Single-span deck truss bridge	California	2012	Caltrans (2012)

Note: Ref. = reference.

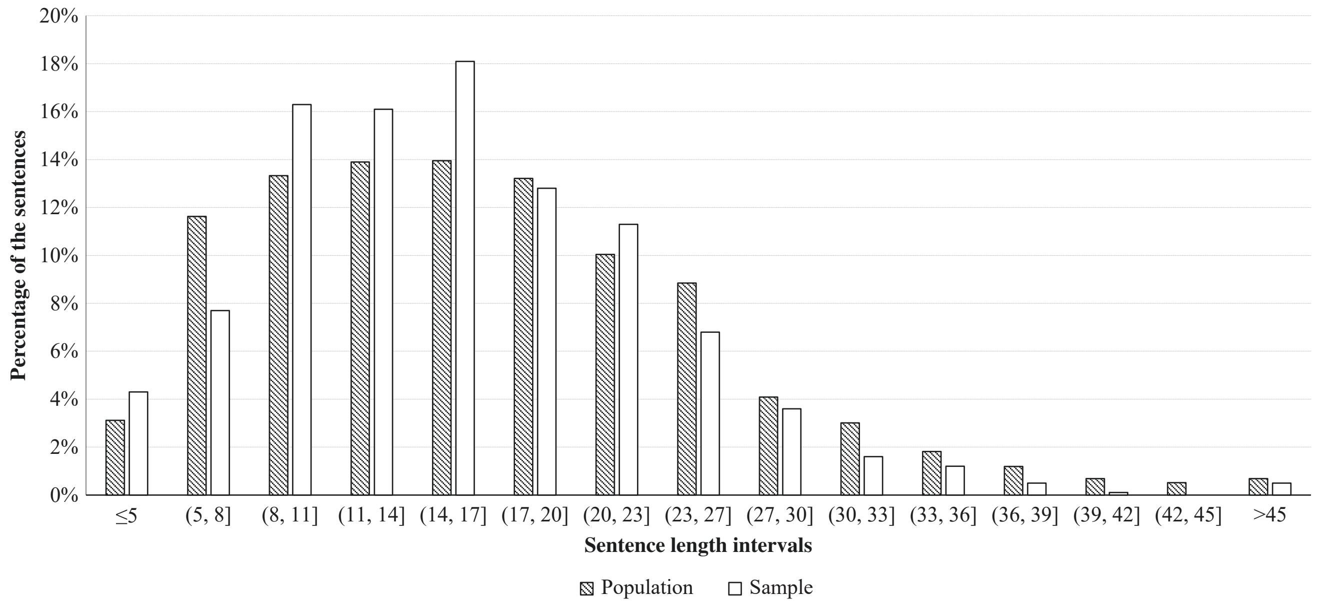


Fig. 7. Distributions of sentence lengths for selected sentences (sample) and for all sentences in 10 bridge inspection reports (population).

indicate that the sample is representative. The sentences were further randomly split into three sets at a ratio of 2:1:1—a training set for algorithm training, a validation set for hyperparameter tuning and algorithm validation, and a testing set for testing the fine-tuned model. As noted previously, 50% of the data were used for training, in order to keep the remaining portion of the data for validation and testing. In the initial method development efforts, the authors also tested the use of 75% of the data for training, which only marginally changed the parsing performance. This indicates that the increase in the ratio of training data, beyond 50%, does not have a substantial impact on the performance results. Table 4 provides a set of sentence examples. Text preprocessing aimed to transform the raw text (the selected sentences) into the format required for dependency relation extraction. Tokenization was used to break down a continuous sentence into a sequence of tokens (e.g., words, digits, punctuations, and whitespaces). Human annotation aimed to mark up the entire data set with gold standard dependency relations. Following the universal dependencies guideline (Marneffe et al. 2014), the sentences were separately annotated by three annotators, with background in both civil engineering and natural language processing. The initial interannotator agreement rate was 87.4% in F-1 measure, which indicates the reliability of the gold standard annotation (Pestian et al. 2012). The discrepancies across the three annotations were then discussed and resolved [according to the universal dependencies guideline (Marneffe et al. 2014)] to reach consensus, thereby achieving a final gold standard annotation with full interannotator agreement.

Feature Extraction and Representation

The training, validation, and testing configurations were first generated from the annotated training, validation, and testing sentences, respectively, using the transition-based DP model. The configuration-based features were extracted based on the defined element positions at the configurations. Second, the syntactic and semantic text features were extracted to represent these configuration-based features. The POS tag set from the Penn Treebank project was used. The tags were analyzed and extracted using the commonly used natural language tool kit (NLTK) POS tagger (Bird et al. 2009). The defined semantic classes were analyzed and extracted using the ontology-based, semisupervised conditional random fields–based named entity recognition (NER) method (Liu and El-Gohary 2017). In this method, the bridge deterioration knowledge ontology (Liu and El-Gohary 2016), which represents bridge deterioration and maintenance knowledge, is used to facilitate the extraction based on content and domain-specific meaning. The errors in the semantic classes were manually checked and corrected by the authors, for evaluation purpose only. The rationale was to separate the errors coming from the NER method versus the errors coming directly from the DP method, in order to separately understand the limit of the proposed semantic feature representation in improving the performance of the DP method. The proposed DP method itself is fully automated and does not require manual interventions in its application. Finally, the extracted text features—words, POS tags, and semantic classes—were represented by the distributed feature representations with a commonly used vector size of 50, using the NN-based hierarchical softmax skip-gram algorithm (Mikolov et al. 2013). This algorithm was selected because it achieved the state-of-the-art performance in distributed feature representation and has been widely applied for supporting many natural language processing tasks inside (e.g., Zhou and El-Gohary 2015) and outside (e.g., Chen and Manning 2014) of the civil engineering domain.

Algorithm Training

The algorithm training aimed to learn the weight vectors for the constituent NN classifiers and the combiner SVM classifier. The training included three main steps. First, all the training configurations were sampled into the defined configuration clusters based on Eq. (1). Second, the four constituent NN classifiers were developed. Each constituent classifier corresponded to a cluster and was trained using the configurations and their GSTs of the cluster. To learn the weights for the NN classifiers [as per Eq. (4)], the backpropagation algorithm (Rumelhart et al. 1986) was used. Third, a combiner SVM classifier was developed. The combiner classifier was trained using all the training configurations and their GSTs. During the training, each of the configurations was represented

Table 4. Examples of sentences in the bridge inspection reports

Report No.[a]	Sentence No.	Original sentence from bridge inspection report
1	1	The one-half inch thick, oil and stone surface treatment, over two inches of bituminous materials, over a corrugated steel deck, still shows full width transverse cracking, open a maximum of one inch, mainly in the areas of the deck, adjacent to the pier.
1	2	The outside fascia deck edge plates still show light to moderate rusting, along their edges.
2	3	Several of the anchor bolts for the cross girder bearings on the pier columns exhibit deficiencies that include mis-drilled holes, bent anchor bolts, improperly installed anchor bolts, and loose nuts.
2	4	The curb faces on the westbound deck have minor widespread spalling.
3	5	Throughout the bridge, the bolted field splices for the deck exhibited isolated instances of loose bolts, missing nuts, and missing bolts (see photo 15)[b].
3	6	Rodents, rodent's dens, and moderate rodent debris were noted in tiers 23–25 of both towers.
4	7	The Pier 1 expansion bearing assemblies exhibited approximately 25 percent loss of protective coating with moderate corrosion and negligible loss of section on the exposed areas.
4	8	The timber deck members were coated with creosote and tar.
5	9	Truss bottom chord members typically have deterioration with section loss at the gusset plates and some surface rust throughout webs and top flanges.
5	10	Minor corrosion and section loss of bottom flange angles.
6	11	The underside of the cap beam between columns C1 and C2 exhibits 3′ × 20″ × 3″ deep spall with two main rebars exposed and one stirrup exposed, 32″ × 16″ × 2″ deep spall and hollow sounding concrete areas 12″ × 18″.
6	12	The joint seal has detached from the joint.
7	13	The paint system of this section appears to be in fair condition overall, with failure on approximately 20 percent of the surface area.
7	14	Some of the rivet heads in these locations have also suffered some moderate section loss.
8	15	South abutment settled downward and retaining walls of abutment rotated outward, allowing span between abutment and bent 2 to settle as well during earthquake.
8	16	Large spall 18″ × 24″ on west wall of north abutment.
9	17	Several anchor bolts and many keeper plates were noted to be missing at the abutment bearings and a steel bolster Girder 2 exhibits section loss.
9	18	The abutment rocker bearings exhibit pack rust between the masonry plate and the rocker.
10	19	At connections there was generally minor crevice corrosion between the eyebar heads and the pin with an average section loss of approximately 1/8″ around the interior circumference.
10	20	Significant section loss to the bottom lacing was found in spots along the top chord.

[a]The report number follows that defined in Table 3.
[b]"Photo 15" in this sentence refers to the photo indexed as 15 in the report of LaDOTD (2008).

with the probability vector (as per Fig. 2). To learn the weight vector of the combiner SVM classifier, the stochastic gradient descent algorithm was used.

Evaluation

The evaluation included algorithm validation and testing. Algorithm validation was conducted, using the configurations, to: (1) select the hyperparameter values for the classifiers, (2) select the feature representation, and (3) compare the performance of the proposed DP algorithm to those of the three baselines—semantic single classifier–based algorithms that used an NN or SVM classifier and a semantic stacked generalization–based algorithm that used cross-validation partitioning for sampling the configurations. The selection and comparison were conducted based on configuration-based accuracy, which is the ratio of the number of correctly classified configurations to the total number of configurations, as per Eq. (5). Algorithm testing was conducted, using the testing sentences, to evaluate the performance of the proposed DP algorithm (with the selected hyperparameters and feature representation) in extracting dependency relations from bridge inspection reports for representing the information about bridge conditions and maintenance actions in a semantically rich structured way. The performance was measured in terms of precision, recall, and F-1 measure, at both the SIE and SIS levels. Precision, as per Eq. (6), is the ratio of the number of correctly extracted SIEs/SISs to the total number of extracted SIEs/SISs. Recall, as per Eq. (7), is the ratio of the number of correctly extracted SIEs/SISs to the total number of SIEs/SISs that should be extracted. F-1 measure, as per Eq. (8), is the weighted harmonic mean of precision and recall.

Threefold cross-validations were performed, during the algorithm validation and testing, to better evaluate the performances of the algorithms. For the cross-validations, the entire data set was randomly split three times, each time into three sets (50% training, 25% validation, and 25% testing). Same as studies for similar applications (e.g., Dhillon et al. 2012; Adriani et al. 2020), the number of folds in this research was set to three to keep more data for testing in each fold. The confidence intervals of the mean values for the evaluation measures were also calculated to evaluate the sensitivity of the performance results. The confidence intervals were calculated using Eq. (9), where $\bar{x}$ is the mean, σ is the standard deviation, n is the number of sentences or configurations in the validation or testing set, z^* is the critical value, and $z^* \frac{\sigma}{\sqrt{n}}$ is the margin of error. At 95% confidence level, $z^* = 1.96$. Because prediction accuracies, precisions, and recalls generally follow a normal distribution (Mirza et al. 2007; Lu et al. 2005), such a distribution was assumed and used for calculating the confidence intervals

$$\text{Configuration-based accuracy} = \frac{\text{number of correctly classified configurations}}{\text{number of all the configurations}} \tag{5}$$

$$\text{Precision} = \frac{\text{number of correctly extracted SIEs (or SISs)}}{\text{number of extracted SIEs (or SISs)}} \tag{6}$$

$$\text{Recall} = \frac{\text{number of correctly extracted SIEs (or SISs)}}{\text{number of SIEs (or SISs) that should be extracted}} \tag{7}$$

$$\text{F-1 measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{8}$$

$$\text{Confidence interval (CI)} = \left(\bar{x} - z^* \frac{\sigma}{\sqrt{n}}, \bar{x} + z^* \frac{\sigma}{\sqrt{n}}\right) \quad (9)$$

Experiments, Results, and Discussion

Hyperparameter Value Selection

The hyperparameter values for the NN and SVM classifiers were selected. Because the activation and the kernel functions are especially important for the constituent NN and the combiner SVM classifiers to collectively capture the nonlinearity of the configurations, combinations of the two types of functions were tested. Four commonly used activation functions (identity, Gaussian, hyperbolic tan, and logistic sigmoid) and four commonly used kernel functions (linear, polynomial, radial basis function, and sigmoid) were tested, resulting in a total of 16 combinations. The selected values are summarized in Table 5.

Feature Representation Selection

Seven text feature representations were tested and compared to investigate the effectiveness of different types of representations. These representations include combinations of the three types of features: words, POS tags, and semantic classes, as shown in Fig. 8. To study the significance levels of the performance differences across these representations, a set of Welch's unequal variances t-tests was conducted. The probability values (p-values) were used to interpret the t-test results: if the p-value is greater than 0.05, there is no significant difference; otherwise the difference is significant. Fig. 8 summarizes the mean configuration-based accuracies, their corresponding confidence intervals, and the p-values for comparing the proposed feature representation to the remaining six representations. The experimental results show that the proposed semantic distributed feature representation—which uses words, POS tags, and semantic classes (FR7 in Fig. 8)—achieved the highest configuration-based accuracy of 91.3% on the testing set (the accuracy might be lower if uncorrected semantic classes were used). By combining the high and low cooccurrence rate features, as well as the syntactic and semantic features, it was effective in capturing the highly variable patterns of the text in the bridge inspection reports.

The feature representations (FR1, FR2, FR3, and FR6) that used words, POS tags, semantic classes, and the combination of POS tags and semantic classes achieved an accuracy of 87.0%, 83.4%, 63.8%, and 84.0%, which is 4.3%, 7.9%, 27.5%, and 7.3% lower compared to the highest (FR7), respectively, on the testing set, with all differences being significant. The lower performance was caused by two main reasons. First, the representations with a low cooccurrence rate (such as FR1) generated too many unseen feature patterns in the testing configurations that have not been learned from the training configurations. Second, the representations with a high cooccurrence rate (such as FR2, FR3, and FR6) caused the configurations that belong to different transition types to have similar and/or identical feature patterns. The unseen and similar/identical feature patterns made the dependency parsing (DP) algorithm limited in effectively distinguishing the configurations of different transition types and therefore resulted in a lower accuracy.

The feature representations (FR4 and FR5) that used the combination of words and POS tags and the combination of words and semantic classes achieved an accuracy of 86.8% and 85.8%, which is 4.5% and 5.5% lower compared to the highest (FR7), respectively, on the testing set, with all differences being significant. The improved performance of FR7, compared to FR4 and FR5, was mainly due to the fact that, in addition to combining the low and high cooccurrence rate features, it also utilized the POS tags and semantic classes jointly. These two types of features are complementary to each other and therefore led to the optimal DP performance. The semantic class features are effective in capturing the dependency relations (word-to-word interactions) between the concepts that have defined semantics. For example, the words *severe* and *corrosion* can be classified into a correct transition type based on their defined semantic meanings: a categorical severity measure (*severe*, as a modifier) describes a deficiency (*corrosion*, as a head). On the other hand, the POS tag features are effective in capturing the relations between the concepts that do not have defined semantics (low-content-bearing words, such as *of*, *on*, and *at*). For example, the SIEs *wearing surface* (as a head) and *concrete deck* (as a modifier) in the phrase *wearing surface on the concrete deck* can be associated with a correct SIE-to-SIE dependency relation based on the POS tag of the *on* (i.e., preposition).

Comparison to Baseline Algorithms

Three baseline DP algorithms were developed for comparative evaluation: a semantic NN-based, a semantic SVM-based, and a semantic stacked generalization (SG)–based. The first two were used to evaluate the effectiveness of the proposed ensemble learning–based approach. The semantic NN-based DP baseline was selected because it is one of the state-of-the-art NN-based DP methods (e.g., Chen and Manning 2014) that has been commonly used as a benchmark (e.g., Weiss et al. 2015 and Alberti et al. 2015). The semantic SVM-based DP baseline was selected because it is commonly used in the literature (e.g., Kudo and Matsumoto 2002; Yamada and Matsumoto 2003). For these baselines, a single NN or SVM classifier was used. The hyperparameter values of the classifiers are given in Table 5. The third baseline was used to evaluate the effectiveness of the proposed sampling approach. It was selected because it is the most similar to the proposed algorithm—except that the proposed algorithm used the similarity-based sampling method rather than cross-validation partitioning. For the SG-based and the proposed DP algorithms, four constituent NN classifiers with the logistics sigmoid activation function and a combiner SVM classifier with the linear kernel functions were used. These functions were selected based on the results in the "Hyperparameter Value Selection" section. All the DP algorithms (for both the proposed and the baseline models) were developed using the proposed semantic distributed feature representation, as per Fig. 3.

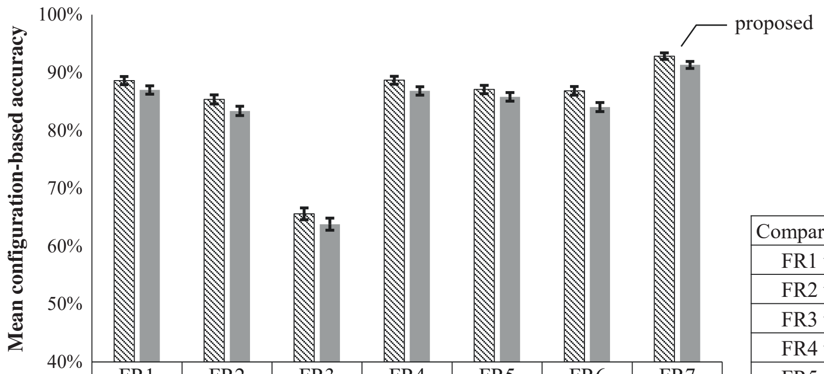
The performances of the proposed and the baseline algorithms are summarized in Fig. 9, and their confusion matrices are shown in Fig. 10. As shown in Fig. 9, the proposed semantic NNE-based DP algorithm (A4) achieved the highest accuracy of 91.3% on the testing set. The semantic NN-based DP baseline with the hyperbolic tan activation function (A2-4) achieved an accuracy of 77.8%. The semantic SVM-based DP baseline with the radial basis function kernel (A1-3) achieved an accuracy of 63.0%. And the semantic SG-based DP algorithm (A3) achieved an accuracy of 76.4%. As shown in Fig. 10, the proposed algorithm achieved improved precisions and recalls across all transition types.

These results indicate that the proposed ensemble learning–based DP approach is effective in dealing with domain-specific and highly technical text (such as that in the bridge inspection reports) for extracting dependency relations. It significantly improved the accuracy by 13.5% and 28.3%, compared to the NN- and SVM-based DP baselines, respectively. This is because,

Table 5. Hyperparameter values of the proposed and the baseline dependency parsing algorithms

Classifier type	Hyperparameter	Value	Explanation
Neural network classifier	Number of network layers[a]	3	This value was selected based on Chen and Manning (2014), because it balances classification accuracy and computational efficiency.
	Hidden layer size[a]	200	—[b]
	Regularization parameter[a]	10^{-8}	—[b]
	Activation function	Logistic sigmoid function or hyperbolic tan function	The combination that used the logistic sigmoid activation function and the linear kernel function achieved the highest configuration-based accuracy on both the validation and testing sets, compared to the other combinations. The logistic sigmoid function was therefore selected for the constituent neural network classifiers. Four commonly used activation functions, including the logistic sigmoid, identity, Gaussian, and hyperbolic tan functions, were tested. The hyperbolic tan function was selected over the other three for the single neural network classifier, because it achieved the highest configuration-based accuracy on both the validation and testing sets.
Support vector machine classifier	Soft margin constant	200 or 1	A set of values, including 1 and those ranging from 20 to 300 with a step size of 20, were tested. A value of 200 for the combiner classifier and a value of 1 for the single classifier were selected to control the margin of the decision boundaries, because they achieved the highest configuration-based accuracy, on the respective validation and testing sets.
	Kernel function	Linear kernel or radial basis function kernel	The combination that used the logistic sigmoid activation function and the linear kernel function achieved the highest configuration-based accuracy on both the validation and testing sets, compared to the other combinations. The linear kernel function was therefore selected for the combiner support vector machine classifier. Four commonly used kernels, including the linear, polynomial, radial basis function, and sigmoid kernels, were tested. The radial basis function kernel was selected over the other three for the single support vector machine classifier, because it achieved the highest configuration-based accuracy on both the validation and testing sets.
	Degree of the polynomial kernel[c]	2	This value was selected because it is enough to capture the nonlinear relationships between features (Ben-Hur and Weston 2010).
	Coefficient of the polynomial and sigmoid kernels[c]	1	This value was selected, because it balances the influence of higher-order terms and that of lower-order terms in the polynomial and sigmoid functions and is commonly used in practice (Ben-Hur et al. 2008).
	Gamma of the radial basis function, polynomial, and sigmoid kernels[c]	$1/n$	This value was set to the inverse of the number of features (i.e., n), which is the commonly used value in SVM (Chang and Lin 2011) to control the curvature of the decision boundaries for preventing overfitting.

[a]The constituent and the single neural network classifiers used the same value.
[b]The explanation is the same as that in the first row of the table.
[c]The combiner and the single support vector machine classifiers used the same value.



	FR1	FR2	FR3	FR4	FR5	FR6	FR7
▨ Validation set	88.6%	85.4%	65.6%	88.7%	87.1%	86.9%	92.8%
■ Testing set	87.0%	83.4%	63.8%	86.8%	85.8%	84.0%	91.3%

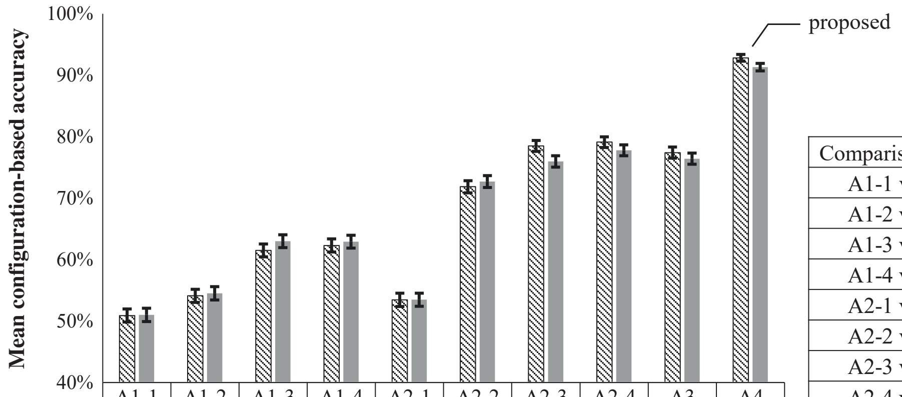
Feature representations

Comparison group	P-value
FR1 vs. FR7	0.0000
FR2 vs. FR7	0.0000
FR3 vs. FR7	0.0011
FR4 vs. FR7	0.0000
FR5 vs. FR7	0.0000
FR6 vs. FR7	0.0001
FR7 vs. FR7	1.0000

P-values

- FR1 = words; FR2 = POS tags; FR3 = semantic classes; FR4 = words + POS tags; FR5 = words + semantic classes; FR6 = POS tags + semantic classes; FR7 = words + POS tags + semantic classes.
- The "I-shape" bar indicates the confidence interval calculated from the corresponding mean configuration-based accuracy.
- The p-values were calculated from the Welch's unequal variances t-tests (using the testing set) and are significant at 0.05 level (2-tailed).

Fig. 8. Performance results for feature representation selection.



	A1-1	A1-2	A1-3	A1-4	A2-1	A2-2	A2-3	A2-4	A3	A4
▨ Validation set	50.9%	54.1%	61.5%	62.3%	53.4%	71.9%	78.5%	79.1%	77.4%	92.8%
■ Testing set	51.0%	54.5%	63.0%	62.9%	53.5%	72.7%	76.0%	77.8%	76.4%	91.3%

Dependency parsing (DP) algorithms

Comparison group	P-value
A1-1 vs. A4	0.0014
A1-2 vs. A4	0.0000
A1-3 vs. A4	0.0310
A1-4 vs. A4	0.0071
A2-1 vs. A4	0.0025
A2-2 vs. A4	0.0006
A2-3 vs. A4	0.0000
A2-4 vs. A4	0.0019
A3 vs. A4	0.0000
A4 vs. A4	1.0000

P-values

- A1-1, A1-2, A1-3, and A1-4 are semantic support vector machine-based DP algorithms with linear, polynomial, radial basis function (RBF), and sigmoid kernel functions, respectively.
- A2-1, A2-2, A2-3, and A2-4 are semantic neural network-based DP algorithms with Gaussian, identity, logistic sigmoid, and hyperbolic tan activation functions, respectively.
- A3 is the semantic stacked generalization-based DP algorithm; A4 is the proposed semantic neural network ensemble-based DP algorithm.
- The "I-shape" bar indicates the confidence interval calculated from the corresponding mean configuration-based accuracy.
- The p-values were calculated from the Welch's unequal variances t-tests (using the testing set) and are significant at 0.05 level (2-tailed).

Fig. 9. Performance results for proposed and baseline dependency parsing algorithms.

by using the constituent and combiner classifiers, the proposed ensemble learning–based DP approach was able to sufficiently capture the distributions of all the configurations, which were too complex to be captured by a single classifier. The results also indicate that the proposed similarity-based sampling method is effective in capturing the complex configuration distributions of the text. It significantly improved the accuracy by 14.9% compared to the SG-based DP baseline. This is because the sampling method used the similarities measured in multiple feature spaces as a collective criterion to sample the configurations into meaningful clusters (see the "Similarity-Based Sampling" section). Conversely, the SG-based algorithm simply clustered the configurations using cross-validation partitioning. Only learning from the similarly distributed and more easily separable configurations allowed each constituent classifier to sufficiently capture the local distributions of the configurations, which resulted in

Gold standard transition type

Predicted transition type	Shift	Left-arc	Right-arc	Precision
Shift	2497	1283	323	60.9%
Left-arc	374	1787	145	77.5%
Right-arc	240	676	897	49.5%
Recall	80.3%	47.7%	65.7%	**63.0%**

(a)

Gold standard transition type

Predicted transition type	Shift	Left-arc	Right-arc	Precision
Shift	3433	370	300	83.7%
Left-arc	423	1754	129	76.1%
Right-arc	436	169	1208	66.6%
Recall	80.0%	76.5%	73.8%	**77.8%**

(b)

Gold standard transition type

Predicted transition type	Shift	Left-arc	Right-arc	Precision
Shift	3391	458	262	82.6%
Left-arc	369	1823	115	79.1%
Right-arc	560	175	1069	59.0%
Recall	78.4%	74.2%	74.1%	**76.4%**

(c)

Gold standard transition type

Predicted transition type	Shift	Left-arc	Right-arc	Precision
Shift	3819	153	131	93.1%
Left-arc	112	2132	62	92.5%
Right-arc	165	93	1555	85.8%
Recall	93.2%	89.7%	89.0%	**91.3%**

(d)

- The bold font indicates the mean configuration-based accuracy.
- The precisions, recalls, and accuracies were calculated using the testing set.
- SVM = support vector machine; NN = neural network; SG = stacked generalization; NNE = neural network ensemble.

Fig. 10. Confusion matrices for proposed and baseline dependency parsing algorithms: (a) confusion matrix for semantic SVM-based dependency parsing algorithm; (b) confusion matrix for semantic NN-based dependency parsing algorithm; (c) confusion matrix for semantic SG-based dependency parsing algorithm; and (d) confusion matrix for semantic NNE-based dependency parsing algorithm (proposed).

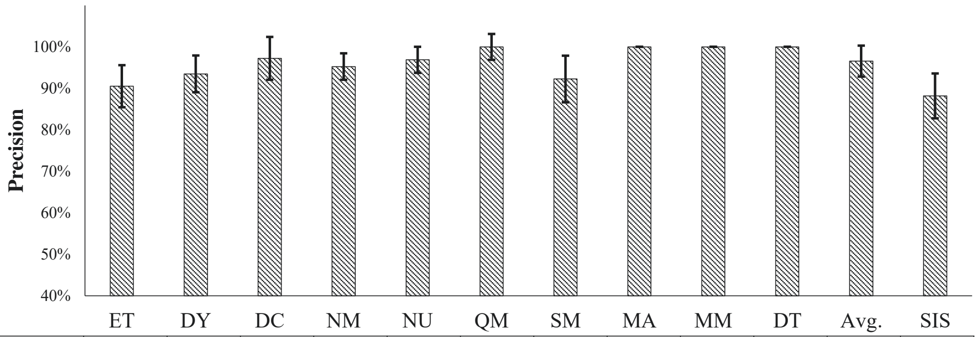
more effective ensembling—improved ability to capture the global distributions of all the configurations.

Performance of the Proposed Dependency Parsing Algorithm

The performance of the proposed semantic NNE-based DP algorithm was evaluated in extracting dependency relations from bridge inspection reports for representing the information about bridge conditions and maintenance actions into SIEs and SISs. The SIE-level measures evaluated how well the individual SIEs can be correctly represented, while the SIS-level measures evaluated how well the SISs can be correctly represented (an SIS representation is correct if and only if all its constituent SIEs are represented correctly). The SIS-level measures are therefore more stringent compared to the SIE-level measures. The experimental results are summarized in Fig. 11. Examples of the extracted information are provided in Fig. 12.

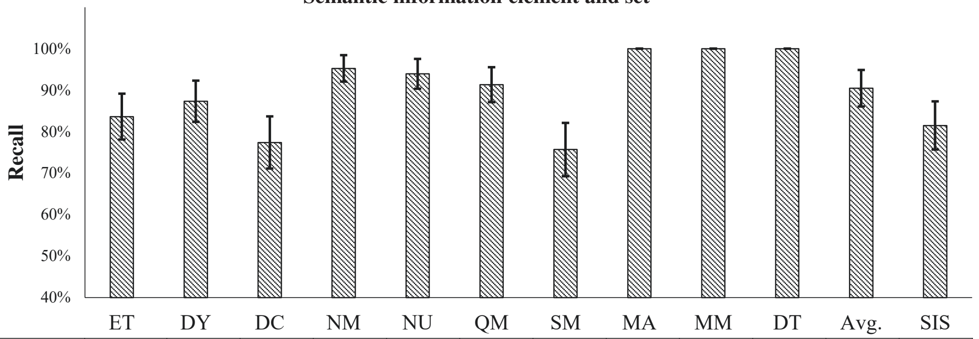
Performance at the Semantic Information Element Level

At the SIE level, on average, the proposed semantic NNE-based DP algorithm achieved a precision, recall, and F-1 measure of 96.6%, 90.4%, and 93.3%, respectively. For some SIE types (e.g., ET, DY, DC, and SM), the algorithm achieved results lower than these averages. For the ET and DY SIEs, it achieved an SIE-level precision, recall, and F-1 measure of 90.5%, 83.6%, and 86.9%, and 93.5%, 87.3%, and 90.3%, which are 6.1%, 6.8%, and 6.4%, and 3.1%, 3.1%, and 3.0% lower compared to the averages, respectively. Two main sources of errors that contributed to these results were identified. First, the large number of the ET and DY SIEs negatively



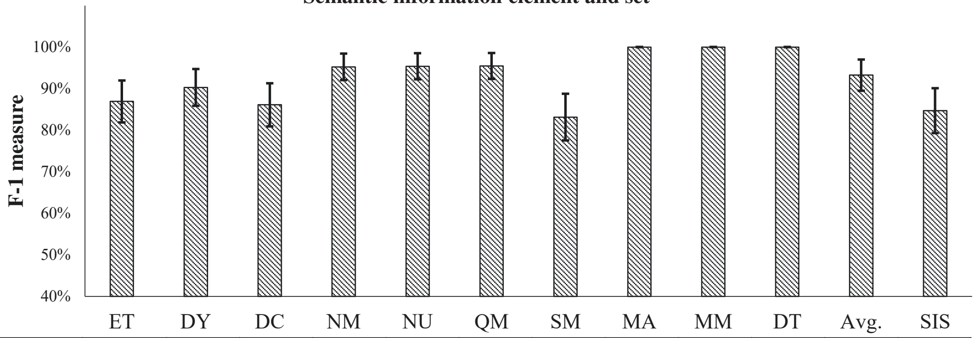
	ET	DY	DC	NM	NU	QM	SM	MA	MM	DT	Avg.	SIS
Prec. (%)	90.5	93.5	97.2	95.2	96.9	100.0	92.3	100.0	100.0	100.0	96.6	88.2
MOE (%)	±5.1	±4.4	±5.2	±3.2	±3.1	±3.1	±5.6	±0.0	±0.0	±0.0	±3.8	±5.4

Semantic information element and set



	ET	DY	DC	NM	NU	QM	SM	MA	MM	DT	Avg.	SIS
Rec. (%)	83.6	87.3	77.4	95.2	93.9	91.3	75.7	100.0	100.0	100.0	90.4	81.5
MOE (%)	±5.5	±5.0	±6.3	±3.2	±3.6	±4.2	±6.4	±0.0	±0.0	±0.0	±4.4	±5.8

Semantic information element and set



	ET	DY	DC	NM	NU	QM	SM	MA	MM	DT	Avg.	SIS
F-1 (%)	86.9	90.3	86.1	95.2	95.4	95.5	83.1	100.0	100.0	100.0	93.3	84.7
MOE (%)	±5.1	±4.4	±5.2	±3.2	±3.1	±3.1	±5.6	±0.0	±0.0	±0.0	±3.8	±5.4

Semantic information element and set

- ET = bridge element; DY = deficiency; DC = deficiency cause; NM = numerical measure; NU = numerical measure unit; QM = categorical quantity measure; SM = categorical severity measure; MA = maintenance action; MM = maintenance material; DT = date.
- Avg. = average semantic information element level performance; SIS = semantic information set level performance.
- Prec. = precision; Rec. = Recall; F-1 = F-1 measure; MOE = margin of error, where a confidence interval = (mean - MOE, mean + MOE).

Fig. 11. Performance results for proposed semantic neural network ensemble–based dependency parsing algorithm at semantic information element (SIE) and semantic information set (SIS) level.

affected the performance of the algorithm. Bridge inspection reports tend to have more descriptions about bridge elements and their deficiencies. For example, in the used data set, 50.2% and 22.1% of the concepts are ET and DY SIEs, respectively. These SIEs are the main sources of the ambiguities in the dependency relations (i.e., associating the right DY elements to the right ET elements is challenging, given the existences of multiple such SIEs in a sentence). Second, the errors generated during the POS tagging process negatively affected the performance of the algorithm. For example, in the following sentence, the words *shows* and *cut* were incorrectly tagged as *noun* and *verb*, respectively: "The salvaged stringer superstructure, shows flame cut holes for various stringer

Sentence #1	The underside of cap beam between columns C1 and C2, exhibits hollow sounding concrete areas up to 44 " x 36 " hollow sounding concrete, 18 " x 12 " x 3 " deep spall and 2 ' x 8 " x 3 " deep spall with exposed rebars.									
Extraction	**Semantic information set (SIS)**									
	ET	**DY**	**DC**	**NM**	**NU**	**QM**	**SM**	**MA**	**MM**	**DT**
	cap_beam	hollow_sounding_concrete	–	44_36	"_"	–	–	–	–	–
	cap_beam	hollow_sounding_concrete	–	–	–	–	–	–	–	–
	cap_beam	spall	–	18_12_3	"_"_"	–	deep	–	–	–
	cap_beam	spall	–	2_8_3	'_"_"	–	deep	–	–	–
	cap_beam	exposed_rebars	–	2_8_3	'_"_"	–	deep	–	–	–

Sentence #2	The timber deck members were coated with creosote and tar.									
Extraction	**Semantic information set (SIS)**									
	ET	**DY**	**DC**	**NM**	**NU**	**QM**	**SM**	**MA**	**MM**	**DT**
	timber_deck	–	–	–	–	–	–	coated	creosote	–
	timber_deck	–	–	–	–	–	–	coated	tar	–

Sentence #3	Truss diagonals and verticals typically have corrosion and section loss at the lower gusset connections.									
Extraction	**Semantic information set (SIS)**									
	ET	**DY**	**DC**	**NM**	**NU**	**QM**	**SM**	**MA**	**MM**	**DT**
	truss_diagonals	–	corrosion	–	–	–	–	–	–	–
	truss_diagonals	–	section_loss	–	–	–	–	–	–	–
	truss_verticals	–	corrosion	–	–	–	–	–	–	–
	truss_verticals	–	section_loss	–	–	–	–	–	–	–

ET = bridge element; DY = deficiency; DC = deficiency cause; NM = numerical measure; NU = numerical measure unit; QM = categorical quantity measure; SM = categorical severity measure; MA = maintenance action; MM = maintenance material; DT = date.

Fig. 12. Examples of extracted information represented in a semantically rich structured way.

ends, over the east abutment, and along the fascia stringer ends, over each side, of the pier." (ConnDOT 2009). This resulted in incorrectly associating the DY element (*flame cut holes*) to the other ET elements (e.g., *east abutment*, *fascia stringer ends*, and *pier*), instead of correctly associating it to *stringer superstructure*.

For the DC and SM SIEs, the algorithm achieved an SIE-level precision, recall, and F-1 measure of 97.2%, 77.4%, and 86.1%; and 92.3%, 75.7%, and 83.1%, respectively. The recalls of these two SIEs are much lower than the average (13.0% and 14.7% lower for DC and SM, respectively). Two main sources of errors that caused the lower recalls were identified. First, when combining words into SIEs, the information representation method (as per Fig. 5, step 2) sometimes combined multiple DC SIEs into one single element, and therefore led to the low recall for the DC SIEs (multiple DC SIEs should be extracted, while only the single DC element was incorrectly extracted). For example, in the following sentence, the DC SIEs *rodent droppings* and *debris* were combined into one DC element *rodent droppings debris*, which is incorrect: "The interior of the longitudinal box girders exhibited heavy rodent droppings, debris, and nests . . . ." (LaDOTD 2008). For a correct extraction and representation, the three DC SIEs *rodent droppings*, *debris*, and *nests* should all be extracted and represented as separate SIEs. A further analysis revealed the root source of such mistakes: ignoring punctuation during the parsing, which is the default practice according to the universal dependencies guideline (Marneffe et al. 2014) and is commonly applied in other DP methods. Punctuations are in some cases indicative of correct dependency relations. So, when the comma between *rodent droppings* and *debris* was not considered, they were associated with an incorrect dependency relation. Second, the proposed SIE-to-SIE dependency relation types (as per Fig. 4) sometimes limited the information representation, and therefore led to the lower recall of the SM SIEs. For example, in the aforementioned example sentence, the SM SIE *heavy* and the DC SIE *rodent droppings* should be extracted and represented. Although the DP algorithm correctly associated a dependency relation between these two elements, the SM SIE was not represented in an SIS because no semantics (SIE-to-SIE dependency relation types) were defined between the SM and DC SIEs.

Performance at the Semantic Information Set Level

At the SIS level, the proposed semantic NNE-based DP algorithm achieved a precision, recall, and F-1 measure of 88.2%, 81.5%, and 84.7%, respectively. Compared to the average SIE-level measures, the SIS-level precision, recall, and F-1 measure are 8.4%, 8.9%, and 8.6% lower, respectively. This is because (as discussed) the SIS-level measures are naturally more stringent than those at the SIE level. The results also show that the performance in extracting/representing the bridge elements (at the SIE level) sets an upper bound for the entire SIS-level performance. This is because the bridge elements are the root of the extraction and representation, so when a bridge element is extracted and represented incorrectly, its whole SIS becomes incorrect.

Sensitivity Analysis Results

The sensitivity of the accuracy of the proposed semantic NNE-based DP algorithm to different sets of training configurations is low, which indicates that the algorithm generalizes well in classifying the configurations. As shown in Fig. 9, the margin of error is only 0.61%. The sensitivity of the precision and recall of the algorithm to different sets of training sentences is acceptable, which demonstrates the generalizability of the algorithm in representing the information in a semantically rich structured way. As shown in Fig. 11, at the SIE level, on average, the margins of error for the precision, recall, and F-1 measure are 3.8%, 4.4%, and 3.8%, respectively. At the SIS level, the margins of error for the precision, recall, and F-1 measure are 5.4%, 5.8%, and 5.4%, respectively. These margins of error are acceptable. In general, a margin of error within 10% is considered acceptable (Bowles et al. 2001; Zorzi et al. 1998). The errors could be attributed to the challenging characteristics of the text in the bridge inspection reports. Compared to other types of text in the construction domain, such as that in building codes (e.g., Zhou and El-Gohary 2017; Zhang and El-Gohary 2013), the text in the bridge reports is highly nonstandardized and variable, because it is typically written by many different writers/inspectors, who come from various local, state, and federal agencies and have different writing styles. Compared to text from social media, such as Tweets, the text in the reports is highly technical, requiring complex concept identification and relationship association. These unique characteristics add extra challenges to the information extraction and affect the sensitivity of the proposed algorithm. Overall, the proposed semantic NNE-based DP algorithm performed well: it achieved a precision, recall, and F-1 measure of 96.6%, 90.4%, and 93.3% with a margin of error of 3.8%, 4.4%, and 3.8% at the SIE level, and 88.2%, 81.5%, and 84.7% with a margin of error of 5.4%, 5.8%, and 5.4% at the SIS level, respectively.

Limitations

Three limitations of this research are acknowledged. First, the aforementioned error analysis has shown that the errors in the POS tags have negatively affected the performance of the proposed dependency parsing methodology. One main reason for the POS tagging errors is that the NLTK POS tagger, as for all other taggers, was trained using general domain text [e.g., the Wall Street Journal (WSJ) data set]. In their future research, the authors plan to develop a domain-specific POS tagger, and test its impact on the performance of dependency relation extraction. Second, in this research, three types of SIE-to-SIE dependency relations that cover 10 types of SIEs were defined to support the representation of the text. These relation and SIE types were chosen because they are representative of the information needed for better predicting bridge deterioration, yet they are not too abundant or complex to the extent of causing extra errors in the extraction. The experimental results and error analysis, however, revealed that they are sometimes not enough to capture all the needed information. In their future work, the authors plan to explore the use of additional SIEs (e.g., the location of a deficiency) and SIE-to-SIE dependency relations (e.g., ET-DC-SM and ET-DY-Location relations) to identify the optimal number and types of dependency relations. Third, the proposed methodology is limited in dealing with the imbalance in the transition types/classes. As a result, the precision and recall of the majority class (i.e., *shift*) were higher than those of the minority classes (i.e., *left-arc* and *right-arc*). Therefore, for the confusion matrices (Fig. 10), the precision and recall of each individual class and the average accuracy should be interpreted jointly. In their future work, the authors plan to explore the use of data sampling methods (e.g., random oversampling method and synthetic minority oversampling technique), in order to balance the number of configurations in different transition classes for further improving the performance of dependency parsing.

Conclusions, Contributions, and Future Work

In this paper, a novel semantic neural network ensemble (NNE)–based dependency parsing methodology was proposed to extract dependency relations from bridge inspection reports. It automatically links the isolated words into concepts and represents the unstructured and semantically low concepts in a semantically rich structured way that is ready for bridge data analytics. A set of experiments was conducted to evaluate the performance of the proposed dependency parsing algorithm. The experimental results showed that the proposed algorithm achieved an average SIE-level precision, recall, and F-1 measure of 96.6%, 90.4%, and 93.3% with a margin of error of 3.8%, 4.4%, and 3.8%, and an SIS-level precision, recall, and F-1 measure of 88.2%, 81.5%, and 84.7% with a margin of error of 5.4%, 5.8%, and 5.4%, respectively. The experimental results also showed that the proposed semantic NNE-based dependency parsing algorithm was effective. First, the proposed semantic distributed feature representation improved the accuracy by 7.3%, compared to the representation without using the semantic features. Second, the proposed similarity-based sampling method improved the accuracy by 14.9%, compared to the method using cross-validation partitioning. Third, by taking an ensemble learning–based approach, the proposed algorithm improved the accuracy by 20.9%, on average, compared to the baselines using a single classifier.

This research contributes to the body of knowledge in four main ways. First, it offers a way of leveraging domain-specific semantics—as captured by the semantic features—for better supporting the analysis of highly technical and domain-specific text for improved extraction of dependency relations. Second, this research offers a new sampling method that utilizes similarities measured in multiple feature spaces as a collective criterion to sample data into meaningful clusters for better supporting ensemble learning. The proposed method allows for generating meaningful clusters that contain the densely and sparsely distributed as well as the correctly and incorrectly densely distributed data. Third, this research provides a novel parsing approach that is semantic, NN-based, and ensemble learning–based. It uses a set of constituent NN classifiers and a combiner SVM classifier to collectively capture the complex distributions of data instances. It was therefore able to provide better parsing performance than that achieved by conventional dependency parsing methods, which only rely on a single classifier. Although the experimental results focused on dependency parsing, the applicability of the method is not limited to this case. Rather, it is a generic machine learning approach, which has the potential to support many other data-driven applications, such as text classification and sentiment analysis. When applying the proposed method to a different knowledge domain or application, one can choose to use another type of constituent or combiner classifier and test if the classifier of choice can improve the performance for the application at hand. Fourth, and most importantly, this research offers an automated method to extract word-to-word dependency relations from bridge inspection reports. It automatically links isolated words into concepts and represents the unstructured and semantically low concepts in a semantically rich structured way that is ready to be used in data analytics for predicting bridge deterioration. The proposed method would therefore allow the use of untapped wealth of data in the unstructured reports for improved bridge deterioration prediction and enhanced maintenance decision making.

In their future work, the authors will explore additional ways to improve dependency parsing methods to further support relation extraction in the domain, with a focus on three main directions. First, further validating the proposed semantic NNE-based dependency parsing method on a larger collection of bridge inspection reports, as well as on other types of reports such as bridge maintenance reports and bridge accident reports. Second, investigating different types of neural network architectures (e.g., LSTM) and different types of transition-based approaches (e.g., top-down and bottom-up predictions). Third, conducting further error analysis to study the error propagation and see which NER errors propagate into DP errors and which do not. These efforts could further advance our knowledge of how to automatically extract information from highly technical and domain-specific text, thereby offering more avenues to learn from the big data that we possess and leading to more opportunities for improved data-driven decision making.

Data Availability Statement

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 (the bridge inspection reports as per Table 3). Some or all data, models, or code that support the findings of this study are available from the corresponding author upon reasonable request (the Python code developed for the implementation and the experimental testing of the proposed semantic neural network ensemble–based dependency parsing method).

Acknowledgments

The authors would like to thank the National Science Foundation (NSF). This paper is based upon work supported by NSF under Grant No. 1937115. Funding for this research was also provided in part by the National Center for Supercomputing Applications (NCSA) at the University of Illinois at Urbana-Champaign through the NCSA Faculty Fellows program. 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 aforementioned entities.

References

Adriani, M., M. Brambilla, and M. Di Giovanni. 2020. "Extraction of relations between entities from human-generated content on social networks." In Vol. 11609 of *Proc., Current Trends in Web Engineering ICWE 2019*, edited by M. Brambilla, C. Cappiello, and S. Ow. Cham, Switzerland: Springer. https://doi.org/10.1007/978-3-030-51253-8_7.

Alberti, C., D. Weiss, and S. Petrov. 2015. "Improved transition-based parsing and tagging with neural networks." In *Proc., 2015 Conf. on Empirical Methods in Natural Language Processing*, 1354–1359. Stroudsburg, PA: Association for Computational Linguistics.

ASCE. 2017. "2017 Infrastructure report card." Accessed June 15, 2019. https://www.infrastructurereportcard.org.

Attardi, G., and F. Dell'Orletta. 2009. "Reverse revision and linear tree combination for dependency parsing." In *Proc., Human Language Technologies: The 2009 Annual Conf. of the North American Chapter of the Association for Computational Linguistics*, 261–264. Stroudsburg, PA: Association for Computational Linguistics.

Babbar, R., and B. Schölkopf. 2017. "DiSMEC: Distributed sparse machines for extreme multi-label classification." In *Proc., 10th ACM Int. Conf. on Web Search and Data Mining*, 721–729. New York: Association for Computing Machinery. https://doi.org/10.1145/3018661.3018741.

Bansal, M., K. Gimpel, and K. Livescu. 2014. "Tailoring continuous word representations for dependency parsing." In *Proc., 52nd Annual Meeting of the Association for Computational Linguistics*, 809–814. Stroudsburg, PA: Association for Computational Linguistics.

Ben-Hur, A., C. S. Ong, S. Sonnenburg, B. Schölkopf, and G. Rätsch. 2008. "Support vector machines and kernels for computational biology." *PLoS Comput. Biol.* 4 (10): e1000173. https://doi.org/10.1371/journal.pcbi.1000173.

Ben-Hur, A., and J. Weston. 2010. "A user's guide to support vector machines." In *Proc., Data Mining Techniques for the Life Sciences: Methods in Molecular Biology (Methods and Protocols)*. New York: Humana Press. https://doi.org/10.1007/978-1-60327-241-4_13.

Bickel, S., M. Brückner, and T. Scheffer. 2007. "Discriminative learning for differing training and test distributions." In *Proc., 24th Int. Conf. on Machine Learning*, 81–88. New York: Association for Computing Machinery. https://doi.org/10.1145/1273496.1273507.

Bird, S., E. Loper, and E. Klein. 2009. *Natural language processing with Python.* Beijing: O'Reilly Media.

Bowles, G., P. McAllister, and H. Tarbert. 2001. "An assessment of the impact of valuation error on property investment performance measurement." *J. Property Investment Finance* 19 (2): 139–157. https://doi.org/10.1108/14635780110383695.

Breiman, L. 1996. "Bagging predictors." *Mach. Learn.* 24 (2): 123–140. https://doi.org/10.1007/BF00058655.

Bu, G., J. Lee, H. Guan, M. Blumenstein, and Y. C. Loo. 2014. "Development of an integrated method for probabilistic bridge-deterioration modeling." *J. Perform. Constr. Facil.* 28 (2): 330–340. https://doi.org/10.1061/(ASCE)CF.1943-5509.0000421.

Buchholz, S., and E. Marsi. 2006. "CoNLL-X shared task on multilingual dependency parsing." In *Proc., 10th Conf. on Computational Natural Language Learning*, 149–164. Stroudsburg, PA: Association for Computational Linguistics.

Bui, L. T., V. T. Vu, and T. T. H. Dinh. 2018. "A novel evolutionary multi-objective ensemble learning approach for forecasting currency exchange rates." *Data Knowl. Eng.* 114 (Mar): 40–66. https://doi.org/10.1016/j.datak.2017.07.001.

Caltrans (California DOT). 2012. "Bridge inspection report: Capitola crossing deck truss." Accessed January 7, 2019. http://sccrtc.org/wp-content/uploads/2013/04/BridgeWork/Capitola%20Bridge%20Inspection%20Report.pdf.

Chang, C. C., and C. J. Lin. 2011. "LIBSVM: A library for support vector machines." *ACM Trans. Intell. Syst. Technol.* 2 (3): 1–27. https://doi.org/10.1145/1961189.1961199.

Chen, D., and C. Manning. 2014. "A fast and accurate dependency parser using neural networks." In *Proc., 2014 Conf. on Empirical Methods on Natural Language Processing*, 740–750. Stroudsburg, PA: Association for Computational Linguistics.

Chen, W., and M. Zhang. 2015. "Dependency parsing models." In *Semi-supervised dependency parsing*. 1st ed. Singapore: Springer. https://doi.org/10.1007/978-981-287-552-5.

Cheng, H., H. Fang, X. He, J. Gao, and L. Deng. 2016. "Bi-directional attention with agreement for dependency parsing." In *Proc., 2016 Conf. on Empirical Methods on Natural Language Processing*. Stroudsburg, PA: Association for Computational Linguistics.

Choi, J. D., and A. McCallum. 2013. "Transition-based dependency parsing with selectional branching." In *Proc., 51st Annual Meeting of the Association for Computational Linguistics*, 1052–1062. Stroudsburg, PA: Association for Computational Linguistics.

Collins, M. 2003. "Head-driven statistical models for natural language parsing." *Comput. Linguist.* 29 (4): 589637. https://doi.org/10.1162/089120103322753356.

ConnDOT (Connecticut DOT). 2009. "Structure no. 04601 North Bear Hill Rd over Natchaug River Chaplin Indepth Inspection." Accessed January 7, 2019. https://portal.ct.gov/-/media/DOT/documents/dbridgepubs/BSE04601IR20090708INDEPTHpdfpdf.pdf.

Dhillon, P. S., J. Rodu, M. Collins, D. Foster, and L. Ungar. 2012. "Spectral dependency parsing with latent variables." In *Proc., 2012 Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 205–213. Stroudsburg, PA: Association for Computational Linguistics.

Dietterich, T. G. 2000. "Ensemble methods in machine learning." In *Proc., Multiple Classifier Systems MCS 2000*. Berlin: Springer. https://doi.org/10.1007/3-540-45014-9_1.

Dozat, T., and C. D. Manning. 2017. "Deep Biaffine attention for neural dependency parsing." Preprint, submitted March 10, 2017. https://arxiv.org/abs/1611.01734.

Dozat, T., and C. D. Manning. 2018. "Simpler but more accurate semantic dependency parsing." In *Proc., 56th Annual Meeting of the Association for Computational Linguistics*. Stroudsburg, PA: Association for Computational Linguistics.

DRJTBC (Delaware River Joint Toll Bridge Commission). 2016. "2015 Toll bridge annual inspection report." Accessed January 7, 2019. http://www.drjtbc.org/wp-content/uploads/Annual_Bridge_Inspection_Report_Final_2-5-16_reduced.pdf.

Dyer, C., M. Ballesteros, W. Ling, and A. Matthews. 2015 "Transition-based dependency parsing with stack long short-term memory." In *Proc., 53rd Annual Meeting of the Association for Computational Linguistics and the 7th Int. Joint Conf. on Natural Language Processing*, 334–343. Stroudsburg, PA: Association for Computational Linguistics.

Eisner, J. M. 1996. "Three new probabilistic models for dependency parsing: An exploration." In *Proc., 16th Conf. on Computational Linguistic*, 340–345. Stroudsburg, PA: Association for Computational Linguistics. https://doi.org/10.3115/992628.992688.

Elworthy, D. 2000. "A finite state parser with dependency structure output." In *Proc., Int. Workshop on Parsing Technologies*. Stroudsburg, PA: Association for Computational Linguistics.

Estes, A. C. 2011. "Bridge maintenance, safety, management, and life-cycle optimization." *J. Struct. Infrastruct. Eng.* 7 (10): 807. https://doi.org/10.1080/15732479.2011.555376.

FHWA (Federal Highway Administration). 2016. "The LTBP deterioration modeling algorithm: An innovative approach to accurate deterioration modeling." Accessed December 4, 2018. https://www.fhwa.dot.gov/publications/ltbpnews/15073.cfm.

Freund, Y., and R. E. Schapire. 1995. "A desicion-theoretic generalization of on-line learning and an application to boosting." In Vol. 904 of *Computational learning theory. EuroCOLT 1995. Lecture notes in computer science (lecture notes in artificial intelligence)*, edited by P. Vitányi, 23–37. Berlin: Springer. https://doi.org/10.1007/3-540-59119-2_166.

Guo, J., W. Che, D. Yarowsky, H. Wang, and T. Liu. 2015. "Cross-lingual dependency parsing based on distributed representations." In *Proc., 53rd Annual Meeting of the Association for Computational Linguistics and the 7th Int. Joint Conf. on Natural Language Processing*, 1234–1244. Stroudsburg, PA: Association for Computational Linguistics.

Haixiang, G., L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing. 2017. "Learning from class-imbalanced data: Review of methods and applications." *Expert Syst. Appl.* 73 (May): 220–239. https://doi.org/10.1016/j.eswa.2016.12.035.

Hall, J., J. Nilsson, and J. Nivre. 2010. "Single malt or blended? A study in multilingual parser optimization." In Vol. 43 of *Trends in parsing technology. Text, speech and language technology*, edited by H. Bunt, P. Merlo, and J. Nivre, 19–33. Dordrecht, Netherlands: Springer. https://doi.org/10.1007/978-90-481-9352-3_2.

Harispe, S., S. Ranwez, S. Janaqi, and J. Montmain. 2015. "Semantic similarity from natural language and ontology analysis." *Synth. Lect. Hum. Lang. Technol.* 8 (1): 1–254. https://doi.org/10.2200/S00639ED1V01Y201504HLT027.

Hashimoto, K., C. Xiong, Y. Tsuruoka, and R. Socher. 2017. "A joint many-task model: Growing a neural network for multiple NLP tasks." In *Proc., 2017 Conf. on Empirical Methods in Natural Language Processing*. Stroudsburg, PA: Association for Computational Linguistics.

Henderson, J. 2004. "Discriminative training of a neural network statistical parser." In *Proc., 42nd Annual Meeting on Association for Computational Linguistics*. Stroudsburg, PA: Association for Computational Linguistics. https://doi.org/10.3115/1218955.1218968.

Huang, L., and K. Sagae. 2010. "Dynamic programming for linear-time incremental parsing." In *Proc., 48th Annual Meeting of the Association for Computational Linguistics*, 1077–1086. Stroudsburg, PA: Association for Computational Linguistics.

Huang, Y.-H. 2010. "Artificial neural network model of bridge deterioration." *J. Perform. Constr. Facil.* 24 (6): 597–602. https://doi.org/10.1061/(ASCE)CF.1943-5509.0000124.

INDOT (Indiana DOT). 2007. "Bridge inspection report Sherman Minton Bridge I-64 over the Ohio River." Accessed January 7, 2019. https://www.in.gov/indot/projects/files/20071100_Inspection_Report.pdf.

Jacobs, R. A., M. I. Jordan, S. J. Nowlan, and G. E. Hinton. 1991. "Adaptive mixtures of local experts." *Neural Comput.* 3 (1): 79–87. https://doi.org/10.1162/neco.1991.3.1.79.

Kameshwar, S., and J. E. Padgett. 2017. "Characterizing and predicting seismic repair costs for bridges." *J. Bridge Eng.* 22 (11): 04017083. https://doi.org/10.1061/(ASCE)BE.1943-5592.0001129.

Kiperwasser, E., and Y. Goldberg. 2016. "Simple and accurate dependency parsing using bidirectional LSTM feature representations." *Trans. Assoc. Comput. Ling.* 4: 313–327. https://doi.org/10.1162/tacl_a_00101.

Kudo, T., and Y. Matsumoto. 2002. "Japanese dependency analysis using cascaded chunking." In *Proc., 6th Conf. on Natural Language Learning*, 1–7. New York: Association for Computing Machinery. https://doi.org/10.3115/1118853.1118869.

Kuncoro, A., M. Ballesteros, L. Kong, C. Dyer, G. Neubig, and N. A. Smith. 2017. "What do recurrent neural network grammars learn about syntax?" In *Proc., 55th Annual Conf. of the Association for Computational Linguistics*. Stroudsburg, PA: Association for Computational Linguistics.

LaDOTD (Louisiana DOT and Development). 2008. "Inspection of the Hale Boggs Memorial Bridge in Luling-Destrehan." Accessed January 7, 2019. http://www8.dotd.la.gov/Proposals/450-37-0022/Reports%20and%20Notes/Bridge%20Inspection%20Report.pdf.

Liu, H., and S. Madanat. 2014. "Adaptive optimisation methods in system-level bridge management." *Struct. Infrastruct. Eng.* 11 (7): 884–896. https://doi.org/10.1080/15732479.2014.920038.

Liu, K., and N. El-Gohary. 2016. "Semantic modeling of bridge deterioration knowledge for supporting big bridge data analytics." In *Proc., 2016 ASCE Construction Research Congress (CRC)*, 930–939. Reston, VA: ASCE. https://doi.org/10.1061/9780784479827.094.

Liu, K., and N. El-Gohary. 2017. "Ontology-based semi-supervised conditional random fields for automated information extraction from bridge inspection reports." *Autom. Constr.* 81 (Sep): 313–327. https://doi.org/10.1016/j.autcon.2017.02.003.

Lu, D., Y. Qiao, P. A. Dinda, and F. E. Bustamante. 2005. "Characterizing and predicting TCP throughput on the wide area network." In *Proc., 25th IEEE Int. Conf. on Distributed Computing Systems*, 414–424. Piscataway, NJ: IEEE. https://doi.org/10.1109/ICDCS.2005.17.

Marneffe, M., T. Dozat, N. Silveira, K. Haverinen, F. Ginter, J. Nivre, and C. Manning. 2014. "Universal stanford dependencies: A cross-linguistic typology." In *Proc., 9th Int. Conf. on Language Resources and Evaluation*, 4585–4592. Paris: European Languages Resources Association.

Mayberry, M., and R. Miikkulainen. 2005. "Broad-coverage parsing with neural networks." *Neural Process. Lett.* 21 (2): 121–132. https://doi.org/10.1007/s11063-004-3423-4.

McDonald, R., and J. Nivre. 2007. "Characterizing the errors of data-driven dependency parsing models." In *Proc., 2007 Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 122–131. Stroudsburg, PA: Association for Computational Linguistics.

MDT (Montana DOT). 2011. *Bridge inspection report bridge no.: L45025001+00001*. Montana, Helena: MDT.

Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. 2013. "Distributed representations of words and phrases and their compositionality." In *Proc., Neural Information Processing Systems Conf. and Workshops*, 3111–3119. New York: Association for Computing Machinery.

Mirza, M., J. Sommers, P. Barford, and X. Zhu. 2007. "A machine learning approach to TCP throughput prediction." In *Proc., 2007 ACM SIGMETRICS Int. Conf. on Measurement and Modeling of Computer Systems*, 97–108. New York: The Association for Computing Machinery. https://doi.org/10.1145/1254882.1254894.

Mishalani, R., and M. McCord. 2006. "Infrastructure condition assessment, deterioration modeling, and maintenance decision-making:

New contributions for improved management." *J. Infrastruct. Syst.* 12 (3): 145–146. https://doi.org/10.1061/(ASCE)1076-0342(2006)12:3(145).

MnDOT (Minnesota DOT). 2006. *Fracture critical bridge inspection in-depth report: I-35W over the Mississippi River at Minneapolis, Minnesota*. St. Paul: MnDOT.

NASEM (National Academies of Sciences, Engineering, and Medicine). 2015. *Long-term bridge performance committee letter report: February 23, 2016*. Washington, DC: National Academies Press. https://doi.org/10.17226/23456.

Nguyen, D. Q., M. Dras, and M. Johnson. 2017. "A novel neural network model for joint POS tagging and graph-based dependency parsing." In *Proc., 2017 SIGNLL Conf. on Computational Natural Language Learning*. Stroudsburg, PA: Association for Computational Linguistics.

NHDOT (New Hampshire DOT). 2009. "In-depth inspection and condition report for the Portsmouth memorial bridge over the Piscataqua River." Accessed January 7, 2019. https://www.nh.gov/dot/projects/portsmouthkittery/documents/Vol1of2_MemorialBridge.pdf.

Nickless, K., and R. A. Atadero. 2018. "Mechanistic deterioration modeling for bridge design and management." *J. Bridge Eng.* 23 (5): 04018018. https://doi.org/10.1061/(ASCE)BE.1943-5592.0001223.

Nivre, J. 2003. "An efficient algorithm for projective dependency parsing." In *Proc., 8th Int. Workshop on Parsing Technologies*, 149–160. Stroudsburg, PA: Association for Computational Linguistics.

Nivre, J., and R. McDonald. 2008. "Integrating graph-based and transition-based dependency parsers." In *Proc., 2008 Annual Conf. of the Association for Computational Linguistics*, 950–958. Stroudsburg, PA: Association for Computational Linguistics.

NYSDOT (New York State DOT). 2015. "New York State Department of Transpiration Bridge inspection report." Accessed January 7, 2019. https://www.dot.ny.gov/main/business-center/designbuildproject20/repository/2015%20Bridge%20Inspection%20Report%20-%2020151228.pdf.

ODOT (Oregon DOT). 2005. "Draft field inspection and condition report for union street railroad bridge." Accessed January 7, 2019. https://businessdocbox.com/Construction/73274137-D-r-a-f-t-field-inspection-and-condition-report.html.

Oflazer, K. 2003. "Dependency parsing with an extended finite-state approach." *Comput. Linguist.* 29 (4): 515–544. https://doi.org/10.1162/089120103322753338.

Pei, W., T. Ge, and B. Chang. 2015. "An effective neural network model for graph-based dependency parsing." In *Proc., 53rd Annual Meeting of the Association for Computational Linguistics and the 7th Int. Joint Conf. on Natural Language Processing*, 313–322. Stroudsburg, PA: Association for Computational Linguistics.

Pes, B., N. Dessì, and M. Angioni. 2017. "Exploiting the ensemble paradigm for stable feature selection: A case study on high-dimensional genomic data." *Inf. Fusion* 35 (May): 132–147. https://doi.org/10.1016/j.inffus.2016.10.001.

Pestian, J. P., L. Deleger, G. K. Savova, J. W. Dexheimer, and I. Solti. 2012. "Natural language processing: The basics." In Vol. 2 of *Pediatric biomedical informatics: Translational bioinformatics*, edited by J. J. Hutton. Dordrecht, Netherlands: Springer. https://doi.org/10.1007/978-94-007-5149-1_9.

Priya, R., and P. Aruna. 2012. "SVM and neural network based diagnosis of diabetic retinopathy." *Int. J. Comput. Appl.* 41 (1): 6–12. https://doi.org/10.5120/5503-7503.

Raghavan, A., F. Di Troia, and M. Stamp. 2019. "Hidden Markov models with random restarts versus boosting for malware detection." *J. Comput. Virol. Hacking Tech.* 15 (2): 97–107. https://doi.org/10.1007/s11416-018-0322-1.

Ramanathan, K., J. S. Jeon, B. Zakeri, R. DesRoches, and J. E. Padgett. 2015. "Seismic response prediction and modeling considerations for curved and skewed concrete box-girder bridges." *Earthquakes Struct.* 9 (6): 1153–1179. https://doi.org/10.12989/eas.2015.9.6.1153.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams. 1986. "Learning representations by back-propagating errors." *Nature* 323 (6088): 533–536. https://doi.org/10.1038/323533a0.

Sagae, K., and A. Lavie. 2006. "Parser combination by reparsing." In *Proc., Human Language Technology Conf. of the NAACL*, 129–132. Stroudsburg, PA: Association for Computational Linguistics.

Samuelsson, C. 2000. "A statistical theory of dependency syntax." In *Proc., 18th Conf. on Computational Linguistics*, 684–690. Stroudsburg, PA: Association for Computational Linguistics. https://doi.org/10.3115/992730.992745.

Schapire, R. E. 1990. "The strength of weak learnability." *Mach. Learn.* 5 (2): 197–227. https://doi.org/10.1023/A:1022648800760.

Schiele, B. 2002. "How many classifiers do I need?" In *Proc., 16th Int. Conf. on Pattern Recognition*. New York: Institute of Electrical and Electronics Engineers.

Shibuya, N., B. T. Nukala, A. I. Rodriguez, J. Tsay, T. Q. Nguyen, S. Zupancic, and D. Y. C. Lie. 2015. "A real-time fall detection system using a wearable gait analysis sensor and a support vector machine (SVM) classifier." In *Proc., 8th Int. Conf. on Mobile Computing and Ubiquitous Networking*, 66–67. New York: Institute of Electrical and Electronics Engineers. https://doi.org/10.1109/ICMU.2015.7061032.

Strubell, E., and A. McCallum. 2017. "Dependency parsing with dilated iterated graph CNNs." In *Proc., 2nd Workshop on Structured Prediction for Natural Language Processing*. Stroudsburg, PA: Association for Computational Linguistics.

Sun, S. 2013. "A survey of multi-view machine learning." *Neural Comput. Appl.* 23 (7): 2031–2038. https://doi.org/10.1007/s00521-013-1362-6.

Sun, Y., M. S. Kamel, and Y. Wang. 2006. "Boosting for learning multiple classes with imbalanced class distribution." In *Proc., 6th Int. Conf. on Data Mining*, 592–602. New York: Institute of Electrical and Electronics Engineers.

Tapanainen, P., and T. Järvinen. 1997. "A non-projective dependency parser." In *Proc., 5th Conf. on Applied Natural Language Processing*, 61–71. Stroudsburg, PA: Association for Computational Linguistics. https://doi.org/10.3115/974557.974568.

Wang, W., and M. P. Harper. 2004. "A statistical constraint dependency grammar (CDG) parser." In *Proc., Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, 42–49. New York: Association for Computing Machinery.

Weiss, D., C. Alberti, M. Collins, and S. Petrov. 2015. "Structured training for neural network transition-based parsing." In *Proc., 53rd Annual Meeting of the Association for Computational Linguistics and the 7th Int. Joint Conf. on Natural Language Processing*, 323–333. Stroudsburg, PA: Association for Computational Linguistics.

Wolpert, D. H. 1992. "Stacked generalization." *Neural Network* 5 (2): 241–259. https://doi.org/10.1016/S0893-6080(05)80023-1.

WSDOT (Washington State DOT). 2009. "Bridge inspection report: South Park Bridge." Accessed January 7, 2019. https://www.kingcounty.gov/transportation/SouthParkBridge/~/media/transportation/SouthParkBridge/docsIII/SPBInspectionReport.ashx.

Xu, C., D. Tao, and C. Xu. 2013. "A survey on multi-view learning." Preprint, submitted April 20, 2013. http://arxiv.org/abs/1304.5634.

Yamada, H., and Y. Matsumoto. 2003. "Statistical dependency analysis with support vector machines." In *Proc., Int. Workshop on Parsing Technologies*, 195–206. Stroudsburg, PA: Association for Computational Linguistics.

Yazdani, M., and J. Henderson. 2015. "Incremental recurrent neural network dependency parser with search-based discriminative training." In *Proc., 19th Conf. on Computational Language Learning*, 142–152. Stroudsburg, PA: Association for Computational Linguistics.

Yu, Z., Y. Zhang, J. You, C. P. Chen, H. S. Wong, G. Han, and J. Zhang. 2017. "Adaptive semi-supervised classifier ensemble for high dimensional data classification." *IEEE Trans. Cybern.* 99 (2): 1–14. https://doi.org/10.1109/TCYB.2017.2761908.

Zhang, C., and Y. Ma. 2012. "Ensemble learning." In *Ensemble machine learning: Methods and applications*. 1st ed. New York: Springer. https://doi.org/10.1007/978-1-4419-9326-7.

Zhang, J., and N. El-Gohary. 2013. "Semantic NLP-based information extraction from construction regulatory documents for automated compliance checking." *J. Comput. Civ. Eng.* 30 (2): 04015014. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000346.

Zhang, P., X. Zhu, Y. Shi, L. Guo, and X. Wu. 2011. "Robust ensemble learning for mining noisy data streams." *Decis. Support Syst.* 50 (2): 469–479. https://doi.org/10.1016/j.dss.2010.11.004.

Zhang, Y., and S. Clark. 2008. "A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing using beam-search." In *Proc., Conf. on Empirical Methods in Natural Language Processing*, 562–571. Stroudsburg, PA: Association for Computational Linguistics.

Zhang, Y., and J. Nivre. 2011. "Transition-based dependency parsing with rich non-local features." In *Proc., 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 188–193. Stroudsburg, PA: Association for Computational Linguistics.

Zhou, H., Y. Zhang, and S. Huang. 2015. "A neural probabilistic structured-prediction model for transition-based dependency parsing." In *Proc., 53rd Annual Meeting of the Association for Computational Linguistics and the 7th Int. Joint Conf. on Natural Language Processing*, 1213–1222. Stroudsburg, PA: Association for Computational Linguistics.

Zhou, P., and N. El-Gohary. 2015. "Ontology-based multilabel text classification of construction regulatory documents." *J. Comput. Civ. Eng.* 30 (4): 04015058. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000530.

Zhou, P., and N. El-Gohary. 2017. "Ontology-based automated information extraction from building energy conservation codes." *Autom. Constr.* 74 (Feb): 103–117. https://doi.org/10.1016/j.autcon.2016.09.004.

Zorzi, M., R. R. Rao, and L. B. Milstein. 1998. "Error statistics in data transmission over fading channels." *IEEE Trans. Commun.* 46 (11): 1468–1477. https://doi.org/10.1109/26.729391.