Transition-Based Dependency Parsing Exploiting Supertags

Hiroki Ouchi, Kevin Duh, Hiroyuki Shindo, and Yuji Matsumoto

***Abstract*—Lexical information, including surface word form and part-of-speech (POS) information, plays a crucial role when predicting ambiguous dependency relationships in dependency parsing. However, for resolving dependency ambiguities, surface word information may be too sparse, while POS information may be too coarse. Supertags, which are lexical templates that represent rich syntactic information, have been shown to provide effective features at an intermediate level on the coarse-to-fine scale. In this work, we present a supertag design framework that allows us to instantiate various supertag sets based on the dependency structures. Using this framework, we instantiate various supertag sets and utilize them as features in transition-based dependency parsing systems. Performing experiments on the Penn Treebank and Universal Dependencies data sets, we show that our supertags are effective for transition-based parsers in multilingual parsing as well as English parsing. The comparison of the results of the different supertag sets shows that it is crucial to incorporate the head directionality, head labels, and dependent possession information in supertags to improve the parser performance.**

***Index Terms*—Dependency parsing, multilingual dependency parsing, supertags, transition-based dependency parsing.**

I. Introduction

DATA-DRIVEN dependency parsing approaches, which make use of machine learning, have achieved great success in the automatic syntactic analysis of natural language [1]. In data-driven approaches, *transition-based* dependency parsing, which utilizes a deterministic shift-reduce process for structural prediction, has received considerable attention because of its low time complexity and the freedom to design features based on a rich context [2]. In particular, the feature definition is the key to the high performance of transition-based dependency parsers.

As feature representations, lexical information, including surface word form and part-of-speech (POS) information, plays a crucial role when predicting ambiguous dependency relationships. However, as features to resolve dependency ambiguities, the surface information of words is sparse while POS information is coarse. Therefore, it is worthwhile to investigate intermediate representations that exist at a coarser level than the words, yet capture the information necessary to resolve dependency ambiguities [3].

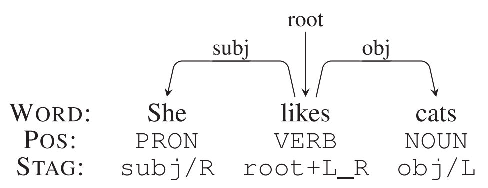


Fig. 1. Illustrative example of supertags for the dependency structure. Supertags encode syntactic information, e.g., the head direction and dependency label.

Recently, **supertags** have been utilized as features in dependency parsing systems [4]. Supertags are tags extended from the notion of POS tags and represent rich syntactic information [5], such as the head direction and dependency label. Fig. 1 illustrates the dependency-based supertags proposed in [4]. In this example, each supertag encodes the head direction with the dependency label and dependent direction.

While supertags can arbitrarily be designed, it is important to keep the adequate balance between the supertag granularity and predictability to improve parsing performance. Increasing the granularity of supertags in order to capture more fine-grained syntactic information results in large tag sets, which tend to be more difficult to predict automatically. In order to improve dependency parsing performance by utilizing supertags, it is necessary to design supertags with the easy automatic assignment that are expressive enough to resolve dependency ambiguities.

In this work, we present a supertag design framework that allows us to design various supertag sets. To investigate the appropriate granularity or design of supertags needed to improve parsing performance, we build various granularity-level supertag sets based on the framework. For English dependency parsing with the supertags, we perform experiments on the Penn Treebank data set. In addition, the utility of the supertags for multilingual dependency parsing is an open question, so we also perform experiments on Universal Dependencies data set (UD; release 1.3).[1] The experimental results show that appropriately designed supertags are effective for dependency parsing.

This work is a substantial extension to the previous work [4]. Based on the supertag design proposed in [4], we formalize a supertag design framework and instantiate various granularity-level supertag sets to investigate which syntactic information should be incorporated into supertags. Using the new supertag sets as features, we perform experiments on multilingual dependency parsing as well as English parsing. The experimental

Manuscript received October 24, 2015; revised June 19, 2016; accepted July 18, 2016. Date of publication August 4, 2016; date of current version August 24, 2016. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Ani Nenkova.

The authors are with the Department of Information Science, Nara Institute of Science and Technology, Ikoma, Nara 630-0101 Japan (e-mail: ouchi.hiroki.nt6@is.naist.jp; kevinduh@is.naist.jp; shindo@is.naist.jp; matsu @is.naist.jp).

Digital Object Identifier 10.1109/TASLP.2016.2598310

[1]http://universaldependencies.org/

results are better than those in [4] thanks to the use of a beam-search perceptron supertagger and parser. To summarize, the main contributions of this work are as follows.

1) We present a supertag design framework.
2) We develop transition-based dependency parsers exploiting various supertag sets.
3) We demonstrate the utility of our supertags for English and multilingual dependency parsing and suggest which syntactic clues should be incorporated into supertags.

II. Related Work

Supertags, which are lexical templates, encode linguistically rich information that imposes complex constraints in a local context [6]. While supertags have been used in frameworks based on lexicalized grammars, e.g., Lexicalized Tree-Adjoining Grammar (LTAG), Head-driven Phrase Structure Grammar (HPSG), and Combinatory Categorial Grammar (CCG), they have scarcely been utilized for dependency parsing so far. As exceptions, Foth *et al.* [7] and Ambati *et al.* [8] have used supertags for dependency parsing.

Foth *et al.* [7] designed supertags based on dependency structure information such as dependency labels and dependents with different levels of granularity. They automatically assigned a single supertag to each word, and the accuracy of automatically assigning their designed supertag set is 67%–84% accurate: the coarsest supertag set (35 tags) is 84.1% and the finest one (12,947 tags) is 67.6%. They then utilized the predicted supertags for dependency parsing and demonstrated that supertags improve German dependency parsing under a Weighted Constraint Dependency Grammar (WCDG), which is not data-driven parsing. In particular, the finest supertag set achieved the biggest improvement in parsing performance (+2.1 points). While they design supertags for WCDG parsing, we explore effective supertag design for data-driven and transition-based dependency parsers.

Ambati *et al.* [8] utilized supertags of the Combinatory Categorial Grammar (CCG) as features for Hindi and English dependency parsers. They reported an improvement of around 0.4 points in UAS using supetag features, and argued that CCG supertags can especially improve long distance dependencies, e.g., coordination and relative clause dependencies. In contrast to their work, we develop a supertag set based on dependency structures because we believe that a supertag design based on dependency structures is more suitable for dependency parsing, rather than one based on another lexicalized grammar formalism.

III. Transition-Based Dependency Parsing

Transition-based approaches are a class of data-driven approaches exploiting machine learning techniques. In this work, we focus on *supervised* methods, which utilize sentences with correct dependency structure annotation as the input for machine learning.

In this framework, transition-based systems derive dependency trees based on a parsing model parameterized over a transition sequence from an initial to some terminal configuration. Given a training set (sentences with dependency structure annotation), a parsing model is induced for parsing a new sentence. Based on the induced model, a transition system, which is an abstract machine consisting of a set of *configurations* and *transitions* between configurations, derives the optimal dependency tree [1].

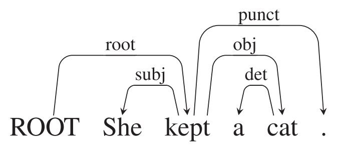
This approach was pioneered by Kudo and Matsumoto [9], Yamada and Matsumoto [10], and Nivre [11] for unlabeled dependency parsing. Nivre *et al.* [12] and Nivre and Scholz [13] extended the approach to labeled dependency parsing. Instead of the greedy search used in the previous systems, beam search was applied to dependency parsing by Zhang and Clark [14], [15]. Of the variations of transition-based systems, **arc-standard** and **arc-eager** are representative systems, and the implementation MaltParser has been widely used so far [16]. In this work, we employ the arc-standard model [17].

In the arc-standard model, configuration $c = (s, b, A)$ consists of a *stack* s, *buffer* b, and set of *dependency arcs* A. The initial configuration for sentence $w_1, \dots, w_n$ is $s = [\texttt{ROOT}]$, $b = [w_1, \dots, w_n]$, and $A = \emptyset$. A configuration c is terminal if the buffer is empty and the stack contains the single node ROOT, and the parse tree is given by A_c. Denoting s_i (i = 1,2,...) as the $i_{\rm th}$ word on the top of the stack, and b_i (i = 1,2,...) as the $i_{\rm th}$ element on the buffer, the arc-standard system defines three types of transitions:

1) LEFT-ARC: adds an arc $s_1 \rightarrow s_2$ and removes s_2 from the stack under the precondition $\mid s \mid \geq 2$.
2) RIGHT-ARC: adds an arc $s_2 \rightarrow s_1$ and removes s_1 from the stack under the precondition $\mid s \mid \geq 2$.
3) SHIFT: moves b_1 from the buffer to the stack under the precondition $\mid b \mid \geq 1$.

For instance, consider the sentence "*She kept a cat.*" in Fig. 2.At step 6, LEFT-ARC is chosen as the next transition. As a result, the second top word on the stack "a" (s_2) depends on the top word "cat" (s_1) and is removed from the stack. At step 7, RIGHT-ARC is chosen as the next transition; hence, the top word on the stack "cat" (s_1) depends on the second top word "kept" (s_2) and is removed from the stack. At step 8, SHIFT is chosen as the next transition, and the first word in the buffer "." (b_1) is removed from the buffer and moved to the stack.

As a result of such transitions, the goal of a transition-based system is to predict a correct transition sequence based on each configuration. Specifically, the system chooses the most probable next transition at each configuration based on *scores*. Those scores are computed as the dot product of the $weight$ and $features$. Features are represented using some lexical information based on the current configuration, such as the word forms and POS tags of some words on the stack/buffer. However, these types of lexical information do not always suffice to predict correct transitions because, as features to resolve dependency ambiguities, surface word information is sparse while POS information is coarse. In this work, we develop and utilize **supertags** as entities at an intermediate level, capturing the information necessary to predict a better transition sequence in transition-based systems.



Step	Transition	Stack	Buffer	A
1	`SHIFT`	[ROOT]	[She kept a cat .]	$\emptyset$
2	`SHIFT`	[ROOT She]	[kept a cat .]	
3	`SHIFT`	[ROOT She kept]	[a cat .]	
4	`LEFT-ARC(subj)`	[ROOT kept]	[a cat .]	$A \cup$ (kept, She)
5	`SHIFT`	[ROOT kept a]	[cat .]	
6	`SHIFT`	[ROOT kept a cat]	[.]	
7	`LEFT-ARC(det)`	[ROOT kept cat]	[.]	$A \cup$ (cat, a)
8	`RIGHT-ARC(obj)`	[ROOT kept]	[.]	$A \cup$ (kept, cat)
9	`SHIFT`	[ROOT kept .]	[]	
10	`LEFT-ARC(punct)`	[ROOT kept]	[]	$A \cup$ (kept, .)
11	`LEFT-ARC(root)`	[ROOT]	[]	$A \cup$ (ROOT, kept)

Fig. 2. Example of a parsing process with the *arc-standard model*.

IV. Supertag Design

The main challenge when designing supertags is to find the right balance between granularity and predictability. Ideally, we would like to increase the granularity of the supertags in order capture finer-grained syntactic information, but large tag sets tend to be more difficult to predict automatically. In this section, we provide a supertag design framework. Then, using the framework, we describe various supertag instantiations.

A. Supertag Design Framework

As Fig. 1 shows, each word in a labeled dependency tree has a head with a dependency label in the left/right direction. Using such syntactic information, we design various supertag sets.

Formally, given the i-th word x_i in a labeled dependency tree y, its supertag $x_i.stag$ is defined as follows:

$$x_i.stag = \textsc{Stag}(x_i, y) \tag{1}$$

where function $\textsc{Stag}(x_i, y)$ can arbitrarily be defined and returns a supertag. In this work, we define the function based on the syntactic information of the head, left dependents, and right dependents as follows:

$$\textsc{Stag}(x_i, y) = \textsc{Head}(x_i, y) \circ \textsc{Dep}(x_i, y) \tag{2}$$

where $\textsc{Head}(x_i, y)$ returns head information, and $\textsc{Dep}(x_i, y)$ returns dependent information. The variability of syntactic granularity can be represented by the definitions of $\textsc{Head}(x_i, y)$ and $\textsc{Dep}(x_i, y)$.

B. Supertag Instantiation

Using the generic supertag design framework, we instantiate a variety of supertag sets. As a basic instantiation, we define the functions in Eq. (2) using the information of *dependency labels* and *head directions* between a word and its head/dependents.

As the head information, we define function $\textsc{Head}(x_i, y)$ as follows:

$$\textsc{Head}(x_i, y) = \textsc{HLabel}(x_i, y) \circ \textsc{Dir}(x_i, x_i.head, y) \tag{3}$$

where $\textsc{HLabel}(x_i, y)$ returns the dependency label of the arc between target word x_i and its head in labeled dependency tree (y), $\textsc{Dir}(x_i, x_i.head, y)$ returns the direction of $x_i.head$ relative to x_i in y, i.e., left (L) or right (R). If a word x_i has "ROOT" as its head, we consider it as having no direction, so $\textsc{Null}$ is returned. Concatenating $\textsc{HLabel}(x_i, y)$ with $\textsc{Dir}(x_i, x_i.head, y)$, head information $\textsc{Head}(x_i, y)$ is defined.

In addition to the head information, we add dependent information by defining function $\textsc{Dep}(x_i, y)$ as follows:

$$\begin{aligned}\textsc{Dep}(x_i, y) = {} & \textsc{hasDep}(x_i, y, L) \circ \textsc{DLabel}(x_i, y, L) \\ & \cup \textsc{hasDep}(x_i, y, R) \circ \textsc{DLabel}(x_i, y, R)\end{aligned} \tag{4}$$

where $\textsc{hasDep}(x_i, y, L/R)$ returns $\textsc{True}_{L/R}$ if a word has any left (L) or right (R) dependents; otherwise, it returns $\textsc{False}$. In addition, $\textsc{DLabel}(x_i, y, L/R)$ returns a set of the dependency labels of obligatory left/right dependents. Here, we define obligatory dependents as dependents which have the following dependency relation labels: "SUB," "OBJ," "PRD," or "VC" in the Penn Treebank, and "nsubj," "nsubjpass," "dobj," "iobj," "csubj," "csubjpass," or "ccomp" in UD.

In previous work on supertag design, Foth *et al.* [7] define $\textsc{Dep}(x_i, y)$ as the function that encodes the order of dependents as well as the dependent labels. However, we do not consider the order to avoid increasing the number of tags. As $\textsc{DLabel}(x_i, y, L/R)$, Ouchi *et al.* [4] limited only the obligatory dependents of verbs. In this work, we take into account all the obligatory dependents and are not limited to only verbs.

Based on Eqs. (3) and (4), we define various granularity-level supertag sets by ablating each function. Table I shows our seven supertag sets. Stag-A is the most basic instantiation with all the functions. Ablating the function DLabel from Stag-A, Stag-B is instantiated, which encodes no dependency labels for obligatory dependents and corresponds to Model 1 proposed by Ouchi *et al.* [4]. Similarly, other supertag sets are instantiated by ablating other functions.

TABLE I
SUPERTAG SETS

	HLabel	Dir	hasDep	DLabel
STAG-A	○	○	○	○
STAG-B	○	○	○	×
STAG-C	○	○	×	○
STAG-D	○	×	×	○
STAG-E	○	○	×	×
STAG-F	×	○	○	×
STAG-G	○	×	×	×

TABLE II
EXAMPLES OF SUPERTAG NOTATIONS

	She	kept	a	cat	.
STAG-A	subj/R	root+subj/L_obj/R	det/R	obj/L	punct/L
STAG-B	subj/R	root+L_R	det/R	obj/L	punct/L
STAG-C	subj/R	root+subj_obj	det/R	obj/L	punct/L
STAG-D	subj	root+subj_obj	det	obj	punct
STAG-E	subj/R	root	det/R	obj/L	punct/L
STAG-F	R	L_R	R	L	L
STAG-G	subj	root	det	obj	punct

C. Supertag Notation

Supertag notations for each supertag set can be defined arbitrarily. As an example, we introduce our notations for each supertag set, shown in Table II.

For instance, consider the word "kept" in the example sentence in Fig. 2. This word has `root+L/subj_R/obj` as its supertag for STAG-A. First, we encode the head information of "kept" using the functions in Eq. (3):

$$\text{HLABEL}(x_i = \text{kept},\ y = y') = root$$

$$\text{DIR}(x_i = \text{kept},\ x_i.head = \text{ROOT},\ y = y') = \text{NULL}$$

where function HLABEL returns the dependency label $root$ on the edge between "ROOT" and "kept," and function DIR returns the head direction NULL. Concatenating these results, we obtain the following head information:

$$\text{HEAD}(x_i = \text{kept},\ y = y') = root \circ \text{NULL} = \texttt{root}$$

where the head information $root \circ$ NULL is converted into the supertag notation as `root`, in which NULL is not literally specified. Note that if the direction is "L" (or "R"), we convert it as `root/L` (or `root/R`).

We then encode dependent information using Eq. (4). The word "kept" has the following dependent information:

$$\text{HASDEP}(x_i = \text{kept},\ y = y',\ L) = \text{TRUE}_L$$

$$\text{HASDEP}(x_i = \text{kept},\ y = y',\ R) = \text{TRUE}_R$$

$$\text{DLABEL}(x_i = \text{kept},\ y = y',\ L) = \{subj\}$$

$$\text{DLABEL}(x_i = \text{kept},\ y = y',\ R) = \{obj\}$$

where function HASDEP returns a boolean variable of $\{\text{TRUE}_{L/R}, \text{FALSE}_{L/R}\}$, and function DLABEL returns a set of dependency labels of all the left/right dependents, which means that, for example, if the target word has two left dependents with the labels "obj" and "prep," DLABEL returns {obj, prep}. Concatenating these results, we obtain the following dependent information:

$$\text{HASDEP}(x_i = \text{kept}, y = y', L) \circ \text{DLABEL}(x_i = \text{kept}, y = y', L) = \text{TRUE}_L \circ \{subj\} = \{subj/L\}$$

$$\text{HASDEP}(x_i = \text{kept}, y = y', R) \circ \text{DLABEL}(x_i = \text{kept}, y = y', R) = \text{TRUE}_R \circ \{obj\} = \{obj/R\}$$

where TRUE_L is converted into "L" and concatenated with $\{subj\}$ into left information set $\{subj/L\}$, and the right dependent information is converted into $\{obj/R\}$ in the same way. Note that, if DLABEL returns $\{obj, prep\}$, the dependent possession information "L" is concatenated with each dependent label, i.e., $\{obj/L, prep/L\}$. We then compute the union of the left and right dependent information sets:

$$\begin{aligned}\{subj/L\} \cup \{obj/R\} &= \{subj/L, obj/R\}\\ &= subj/L_obj/R\end{aligned}$$

where "_" is used to concatenate each element.

Finally, based on Eq. (2), we concatenate the obtained head and dependent information:

$$\begin{aligned}\text{STAG}(x_i = \text{kept},\ y = y') &= \text{root} \circ L/subj_R/obj\\ &= root + L/subj_R/obj\end{aligned}$$

where "+" indicates the boundary of the head and dependent information. Depending on each supertag set, different syntactic information is ablated from STAG-A and encoded for each supertag, as shown in Table II.

V. SUPERTAGGING FOR DEPENDENCY PARSING

To exploit supertags in transition-based dependency parsing systems, we need to automatically assign them to each word. We conduct the automatic assignment of our designed supertags by adopting the same approach used in sequence labeling tasks such as POS tagging.

A. Automatic Supertag Assignment

We build a supertagger to assign supertag y of supertag set Y to the i-th word x_i in a sentence in a left-to-right manner, based on the following equation:

$$\hat{y} = \underset{y \in Y}{\text{argmax}}\ \ \mathbf{w}\ \ \phi(x_i, y) \tag{5}$$

where the score of supertag y is computed by the dot-product of weight vector $\mathbf{w}$ and feature vector ϕ, and the highest scoring supertag $\hat{y}$ is picked for assignment to target word x_i. The weight vector is parameterized using training data. The feature vector is defined using feature templates. We extract features from a 7-word window (the *feature window*) surrounding target word x_i using the feature templates shown in Table IV.

We define *Unigram*, *Bigram*, and *History* feature templates. As the Unigram feature templates, we use the surface word forms and POS tags of the words in the feature window. As the Bigram feature templates, we define conjunctive features by concatenating the surface word form and POS tag information

TABLE III
SUPERTAG STATISTICS

	WSJ-YM	WSJ-ST	UD-AR	UD-DE	UD-ES	UD-ID	UD-RU	UD-ZH	UD-Avg.
STAG-A	**321**	**896**	**998**	**916**	**1209**	**680**	**655**	**630**	**847.50**
STAG-B	79	231	183	227	216	186	233	175	203.33
STAG-C	165	528	522	431	558	370	306	360	424.50
STAG-D	127	412	442	321	445	305	237	300	341.67
STAG-E	21	85	57	64	61	55	74	64	62.50
STAG-F	12	12	12	11	11	11	11	11	11.17
STAG-G	12	49	31	33	32	30	39	38	33.83

TABLE IV
FEATURE TEMPLATES OF SUPERTAGGING MODELS

NAME	FEATURE WINDOW	FEATURE TEMPLATE
Unigram	for p in $x_{i-3}, x_{i-2}, x_{i-1}, x_i, x_{i+1}, x_{i+2}, x_{i+3}$	$\langle p.t \rangle, \langle p.w \rangle$
Bigram	for p, q in $(x_i, x_{i+1}), (x_i, x_{i+2}), (x_i, x_{i+3}), (x_{i-1}, x_i), (x_{i-2}, x_i), (x_{i-3}, x_i), (x_{i+1}, x_{i+2}), (x_{i-2}, x_{i-1})$	$\langle p.t \circ q.t \rangle, \langle p.w \circ q.w \rangle$
History		$\langle x_{i-1}.stag \rangle, \langle x_{i-1}.stag \circ x_i.t \rangle, \langle x_{i-2}.stag \rangle, \langle x_{i-2}.stag \circ x_i.t \rangle, \langle x_{i-2}.stag \circ x_{i-1}.stag \rangle, \langle x_{i-2}.stag \circ x_{i-1}.stag \circ x_i.t \rangle$

NOTATION: Feature conjunction = $\circ$; x_i is the i-th word in the sentence; w = word form; t = POS tag; stag = supertag

of some specific pairs of the words in the feature window. In addition to these feature templates, we dynamically utilize the supertags that have already been predicted during the tagging process as features. When assigning a supertag to target word x_i in the feature window, the previous words, such as x_{i-1} and x_{i-2}, have already been assigned a supertag, which is expected to be helpful for predicting the supertag of the target word. Hence, we define the History feature templates by combining those supertags assigned to x_{i-1} and x_{i-2} with the POS tags or surface word forms. A supertagging model instantiates the features from those feature templates. Using the supertags automatically predicted by the supertagging model, we conduct dependency parsing.

B. Supertag Features for Dependency Parsing

We employ the *arc-standard* model as a transition-based dependency parsing system. Specifically, the system chooses the highest scoring next transition $\hat{t}$ at each configuration c based on the following equation:

$$\hat{t} = \operatorname*{argmax}_{t \in T} \ \mathbf{w} \ \phi(t, c) \tag{6}$$

where the score of transition t is computed by the dot-product of weight vector $\mathbf{w}$ and feature vector ϕ, and the highest scoring transition $\hat{y}$ of the possible transition set T is picked up as the next transition. The weight vector is parameterized using training data. The feature vector is defined with the feature templates shown in Table V. Each feature template is defined with information drawn from the *feature window*, which consists of the top three words (or partial structures) on the stack and the first three words on the buffer.

Unigram, *Bigram*, and *Structural* features are based on the features used in [18] and [19] with some modifications, which we call *Base Features*. In contrast, *Uni-Stag* and *Bi-Stag* are novel feature templates related to supertags, which we call *Supertag Features*.

For the Uni-Stag features ($\langle p.stag \rangle$), we use the supertag of the words p within the feature window. To consider more context, the Bi-Stag features look at pairs of supertags. For some specific pairs (p, q) of the words within the feature window, we set conjunctive features, such as conjunction of the two supertags ($\langle p.stag \circ q.stag \rangle$).

VI. EXPERIMENTS

This section presents the experiments and results for supertagging and transition-based dependency parsing exploiting supertags.

A. Datasets

We performed experiments on English dependency parsing and multilingual dependency parsing.

English Dependency Parsing

For English dependency parsing, we performed experiments on the Wall Street Journal part of the Penn Treebank (PTB) dataset [20]. We converted the constituent trees into two types of dependency format: Yamada and Matsumoto head rules (PTB-YM) [10] using Penn2Malt[2] and Stanford dependencies (PTB-SD) [21] using the converter[3]. We adopted the standard splits, using sections 2–21 for training, section 22 for development, and section 23 for testing. We assigned POS tags to the training data by ten-fold jackknifing, following [19]. The development and test sets were automatically tagged by the tagger trained on the training set.

Multilingual Dependency Parsing

For multilingual dependency parsing, we used the Universal Dependencies (UD; release 1.3) data set [22]. This data set has cross-linguistically consistent treebank annotation for many languages. The annotation scheme is an extension of

[2]http://stp.lingfil.uu.se/ nivre/research/Penn2Malt.jar
[3]http://nlp.stanford.edu/software/stanford-dependencies.shtml

TABLE V
FEATURE TEMPLATES FOR THE *arc-standard* MODEL

NAME	FEATURE WINDOW	FEATURE TEMPLATE
Unigram	for p in $s_0, s_1, s_2, b_0, b_1, b_2$	$\langle p.t\rangle$, $\langle p.w\rangle$, $\langle p.t \circ p.lc.t\rangle$, $\langle p.t \circ p.rc.t\rangle$, $\langle p.t \circ p.lc.t \circ p.rc.t\rangle$
Bigram	for p, q in (s_2, s_1), (s_1, s_0), (s_0, b_0), (b_0, b_1), (b_1, b_2)	$\langle p.t \circ q.t\rangle$, $\langle p.w \circ q.w\rangle$, $\langle p.t \circ q.w\rangle$, $\langle p.w \circ q.t\rangle$, $\langle p.t \circ q.t \circ p.lc.t \circ q.lc.t\rangle$, $\langle p.t \circ q.t \circ p.rc.t \circ q.lc.t\rangle$, $\langle p.t \circ q.t \circ p.lc.t \circ q.rc.t\rangle$, $\langle p.t \circ q.t \circ p.rc.t \circ q.rc.t\rangle$
Structural	for p in $s_0, s_1, s_2, b_0, b_1, b_2$	$\langle dist(p, p.lc) \circ p.t\rangle$, $\langle dist(p, p.rc) \circ p.t\rangle$, $\langle p.nd \circ p.t\rangle$
	for p, q in (s_2, s_1), (s_1, s_0), (s_0, b_0), (b_0, b_1), (b_1, b_2)	$\langle dist(p, q)\rangle$, $\langle dist(p, q) \circ p.t \circ q.t\rangle$
Uni-Stag	for p in $s_0, s_1, s_2, b_0, b_1, b_2$	$\langle p.stag\rangle$
Bi-Stag	for p, q in (s_2, s_1), (s_1, s_0), (s_0, b_0), (b_0, b_1), (b_1, b_2)	$\langle p.stag \circ q.stag\rangle$, $\langle p.stag \circ q.t\rangle$, $\langle p.t \circ q.stag\rangle$, $\langle p.stag \circ q.w\rangle$, $\langle p.w \circ q.stag\rangle$

NOTATION: Feature conjunction = $\circ$; $s_i = i$-th word on the top of the stack; $b_i = i$-th word in the buffer; lc = left-most dependent; rc = right-most dependent; w = word form; t = POS tag; stag = supertag; $dist(p, q)$ = word distance between p and q; $nd = 1$ if the word has no dependent, otherwise 0

the Stanford dependencies [21], [23], [24], Google universal part-of-speech tags [25], and the Interset interlingua for morphosyntactic tagsets [26].

For the target languages, we chose six languages from different language branches: Arabic (AR) from the Semitic languages, German (DE) from the Germanic Languages, Spanish (ES) from the Italic languages, Indonesian (ID) from the Malayo-Polynesian languages, Russian (RU) from the Slavic languages, and Chinese (ZH) from the Sinitic languages. In some language data sets, there are no fine-grained POS tags, and in that case, we used the coarse-grained ones.

B. Supertagging Experimental Setup

To parameterize the supertagging models, we used the averaged perceptron [27] with max violation updates [28]. The number of iterations was to 10. For decoding, we exploited beam search with a beam width of 8. Table III shows the size of each target supertag set.

C. Parsing Experimental Setup

To parameterize the parsing models, we used the averaged perceptron with max violation updates in the same manner as the supertagging experiments. The number of iterations is set to 20. For decoding, we exploited beam search with a beam width of 16. To evaluate the utility of the supertags for *arc-standard* dependency parsers, we used the parsers without supertags as the baseline and compared them with the parsers with supertags.

The supertags used for the parsers were automatically predicted. Following the same procedure as automatic POS tagging, we assigned the proposed supertags to the training data by ten-fold jackknifing. For the development and test data, we automatically assigned the supertags using a supertagger trained on the whole training data.

VII. Results and Analysis

A. Results for Suppertagging

Accuracy of Supertagging

Table VI shows the accuracy of supertagging, which suggests what kind of syntactic information is easy or difficult to predict as sequential labeling.

In all the languages, the results of STAG-F or STAG-G have the highest accuracy. As Table I shows, STAG-F encodes the head directionality and left/right dependent possession information, and STAG-G encodes the dependency label on the edge between the target word and its head. Generally, because smaller tag sets tend to be easier to predict than larger ones, the accuracies of the two supertag sets are higher than others. However, in the six languages of UD, although STAG-F is smaller than STAG-G, the average STAG-G accuracy (88.44% on average for the six UD languages) is higher than the average STAG-F accuracy (87.35%). This suggests that it is not always difficult to predict larger tag sets and, furthermore, the prediction complexity changes according to what information the target tag set encodes. The results for STAG-F suggest that the syntactic information encoded by STAG-F is more difficult to predict as sequential labeling task than the head dependency labels encoded by STAG-G in the majority of languages.

Similarly, regardless of the tag set size, the prediction accuracy of STAG-B (82.45/88.94/88.92% for UD-Avg./PTB-YM/PTB-SD) is lower than STAG-C accuracy (83.42/89.71/89.67% for UD-Avg./PTB-YM/PTB-SD). These two tag sets differ with respect to the encoded syntactic information for dependents. STAG-B encodes the left/right dependent possession information for each target word regardless of whether the dependents are adjunct or core arguments. In contrast, STAG-C encodes the dependency labels only if the dependents are core arguments. This suggests that whether the syntactic information relevant to adjunct arguments is encoded or not cause performance variation. Because STAG-A encodes both the dependent possession and core argument labels as well as the head information, it is more difficult to predict than STAG-B and STAG-C.

STAG-D is built by ablating the head directionality information from STAG-C, so that STAG-D is smaller than STAG-C, which leads to a performance boost relative to STAG-C. Similarly, STAG-E is built by ablating the dependency labels for core arguments from STAG-C and hence encodes only syntactic information relevant to heads. The performance boost relative to STAG-C is also observed. Comparing STAG-D with STAG-E, a noticeable difference in average accuracy is not observed, but the results within each language differ. For instance, in German (UD-DE), the accuracy for STAG-D is higher by over 2.5

TABLE VI
SUPERTAGGING RESULTS

	PTB-YM	PTB-SD	UD-AR	UD-DE	UD-ES	UD-ID	UD-RU	UD-ZH	UD-Avg.
STAG-A	88.05	87.75	79.58	76.17	83.13	80.51	78.74	82.03	80.03
STAG-B	88.94	88.92	82.98	78.47	85.43	82.94	81.10	83.80	82.45
STAG-C	89.71	89.67	83.60	79.78	86.14	84.18	83.22	83.60	83.42
STAG-D	90.77	90.84	85.04	84.89	88.00	86.48	86.20	84.27	85.81
STAG-E	90.56	90.71	87.18	82.25	88.32	87.16	85.73	85.60	86.04
STAG-F	**91.61**	91.81	**89.20**	83.76	89.32	87.99	85.99	**87.85**	87.35
STAG-G	91.60	**91.86**	88.77	**87.45**	**90.31**	**89.24**	**88.60**	86.25	**88.44**

NOTATION : Each number indicates an accuracy, and "UD-Avg." indicates the macro average accuracy for each supertag set over the languages of UD.

points than STAG-E. On the contrary, in Arabic (UD-AR) and Chinese (UD-ZH), the accuracy for STAG-E is higher by around 1-2 points than that for STAG-D. A detailed investigation of this difference is a line of interesting future work.

B. Results for Dependency Parsing With Supertags

In this section, we describe the utility of the supertag features in transition-based dependency parsing systems.

Accuracy of Dependency Parsing With Gold Supertags

The utility of supertag features for dependency parsing changes according to each supertag set and supertagging accuracy. In order to check whether the proposed supertag sets and supertag feature templates capture syntactic information that is helpful for dependency parsing, we performed a parsing simulation experiment in which the condition where an arc-standard parser knows the correct (gold) supertags. In this simulated experiment, the arc-standard model receives the correct supertags and utilizes them as features. Table VII shows the unlabeled attachment scores (UAS) and labeled attachment scores (LAS) of the baseline parsers and parsers with the supertag features.

In English dependency parsing (PTB-MT for Yamada and Matsumoto head rules and PTB-SD for Stanford dependencies), the unlabeled attachment scores of STAG-A/B/F reached around 99%, which indicates that the derived dependency trees were almost perfect. This implies that information provided by the supertags is considerably helpful for the transition-based system to determine the times at which reduce transitions should be conducted. Consider the `RIGHT-ARC` transition, which adds an arc from the second to the top word on the stack and removes the top word from the stack. If there are any words in the buffer that depend on the word on the top of the stack, `RIGHT-ARC` should not be executed. The supertag sets STAG-A/B/F encode the head directionality and dependent possession information, which can implicitly tell the parser in which direction the second-top word in the stack has its head and whether the top word has any dependents in the buffer or not. Because this clue could supplement word form and POS information, a parser was able to select and accumulate the correct local transition under each configuration. In fact, this result suggests that if transition-based systems knew the correct supertags that encode the head directionality and dependent possession information and could use them as features, the dependency parsing problem would be solved almost completely.

In multilingual dependency parsing, although the unlabeled attachment scores were not as high as the ones for English dependency parsing, STAG-A/B/F consistently occupied the top-3 highest UAS rankings over the six languages. The score difference between PTB and UD is likely to be caused by the data size difference, i.e., the data size of PTB is much larger than that of UD, so an investigation of the effect of increasing data size is our interesting future work. In the labeled attachment scores, STAG-F is inferior to STAG-A/B, which is consistent with English dependency parsing. This could be caused by the fact that STAG-F does not encode the head dependency label.

Accuracy of Dependency Parsing With Predicted Supertags

To investigate the utility of our supertag sets in dependency parsing in realistic situations, we performed experiments in which transition-based systems exploited automatically predicted supertags as features. Table VIII shows the unlabeled attachment scores (UAS) and labeled attachment scores (LAS) of the baseline parsers and the parsers with the supertag features.

Overall, the parsers with supertag features outperform the baseline. In particular, across the six languages of UD, a performance boost of the parsers with STAG-B is observed, yielding increases of around 1.0 point in UAS and 1.3 points in LAS. In Spanish (UD-ES) and Indonesian (UD-ID), the biggest improvements were achieved (+1.5 points in UAS and +2.0 points in LAS). In English dependency parsing (PTB-MT, PTB-SD), although the improvements of UAS/LAS were smaller than for the six languages of UD, the supertag features worked effectively. The biggest improvement (+0.34/+0.36 points in UAS/LAS of PTB-MT and +0.44/+57 points in UAS/LAS of PTB-SD) was achieved with STAG-B, which is the same tendency as in the languages of UD.

Comparing the results with the gold and predicted supertags, the predicted supertags of STAG-F were not as effective, although the gold ones were useful for parsing. In English parsing, STAG-F was not effective for improving UAS and LAS. In the six languages of UD, although STAG-F was a little bit effective on average (+0.41 points in UAS and +35 points in LAS), the improvement was the worst of the seven supertags. In contrast, while the gold supertags of STAG-C/D/E/G did not have much predictability compared with STAG-A/F, the predicted supertags of STAG-C/D/E/G achieved almost the same UAS and LAS as STAG-A and were better than STAG-F on average. In particular, for LAS, the other supertag sets outperformed STAG-F (around

TABLE VII
Dependency Parsing Results With Gold Supertags

	PTB-YM	PTB-SD	UD-AR	UD-DE	UD-ES	UD-ID	UD-RU	UD-ZH	UD-Avg.
Baseline	92.60/91.31	92.00/89.33	80.51/74.89	84.53/77.97	86.43/81.62	84.01/78.08	83.37/77.32	83.35/79.48	83.70/78.23
Stag-A	99.06/99.01	98.47/98.27	90.50/89.62	95.03/94.22	95.27/94.87	93.40/92.37	92.69/91.61	96.20/95.42	93.85/93.02
Stag-B	**99.10/99.08**	**98.65/98.55**	**90.95/90.50**	95.71/**95.05**	**95.86/95.65**	94.17/**93.63**	93.74/**93.03**	**96.76/96.27**	94.53/**94.02**
Stag-C	97.77/97.77	96.88/96.78	87.13/86.71	90.52/90.12	93.06/92.93	89.66/89.16	89.38/88.99	95.37/94.97	90.85/90.48
Stag-D	98.44/98.44	96.10/96.00	88.41/88.31	93.62/93.39	94.23/94.23	91.17/91.09	91.02/90.75	95.64/95.47	92.35/92.21
Stag-E	98.45/98.43	96.88/96.76	88.54/88.15	93.36/92.95	94.00/93.86	90.85/90.46	90.85/90.31	95.36/94.88	92.16/91.77
Stag-F	98.77/97.10	**98.65**/95.23	90.67/83.41	**96.78**/88.00	95.52/89.83	**94.44**/87.20	**94.59**/86.64	95.98/91.20	**94.66**/87.71
Stag-G	97.65/97.65	96.17/96.15	86.70/86.63	90.61/90.39	92.88/92.86	89.74/89.61	89.57/89.32	95.41/95.41	90.82/90.70

NOTATION: Each number indicates UAS/LAS, in which "UAS" is the unlabeled attachment score and "LAS" is the labeled attachment score.

TABLE VIII
Dependency Parsing Results With Predicted Supertags

	PTB-YM	PTB-SD	UD-AR	UD-DE	UD-ES	UD-ID	UD-RU	UD-ZH	UD-Avg.
Baseline	92.60/91.31	92.00/89.33	80.51/74.89	84.53/77.97	86.43/81.62	84.01/78.08	83.37/77.32	83.35/79.48	83.70/78.23
Stag-A	92.80/91.61	92.38/89.89	81.26/76.18	85.05/79.09	87.65/83.55	85.26/79.55	83.17/77.57	83.77/79.71	84.36/79.28
Stag-B	**92.94/91.67**	**92.44/89.90**	**81.50/76.33**	85.01/78.96	87.72/83.48	85.43/**79.86**	**83.82/78.15**	**84.33/80.40**	**84.64/79.53**
Stag-C	92.59/91.37	92.23/89.85	81.17/76.22	**85.09**/79.31	**87.86/83.69**	84.96/79.26	83.63/77.98	84.19/80.35	84.48/79.47
Stag-D	92.65/91.47	92.33/89.85	81.13/75.94	84.86/79.09	87.65/83.48	85.13/79.57	83.42/77.76	84.00/80.24	84.37/79.35
Stag-E	92.72/91.55	92.24/89.83	81.00/76.05	85.00/**79.34**	87.50/83.43	**85.55**/79.79	83.42/77.75	83.63/79.89	84.35/79.38
Stag-F	92.48/91.25	92.19/89.50	81.05/75.51	84.76/78.03	87.00/82.19	84.80/78.93	83.78/77.73	83.25/79.10	84.11/78.58
Stag-G	92.51/91.33	92.19/89.72	81.15/76.30	85.04/79.07	**87.86**/83.57	85.09/79.47	83.66/77.92	83.72/80.04	84.42/79.40

NOTATION: Each number indicates UAS/LAS, in which "UAS" is the unlabeled attachment score and "LAS" is the labeled attachment score.

TABLE IX
F_1 Scores According to the Dependency Distances

	PTB-YM	PTB-SD	UD-AR	UD-DE	UD-ES	UD-ID	UD-RU	UD-ZH	UD-Avg.
root	95.15/96.10	93.60/94.52	93.32/93.18	86.28/87.51	85.04/89.05	87.61/89.05	90.58/90.18	79.40/79.20	87.04/88.03
1	96.72/96.87	96.34/96.53	94.27/94.59	92.44/92.84	95.85/96.09	95.09/95.45	93.87/94.16	94.71/94.66	94.37/94.63
2	94.14/94.42	93.86/94.08	80.04/81.54	87.27/88.42	91.99/93.53	84.50/85.93	87.20/87.93	86.64/87.46	86.27/87.47
3-6	90.92/91.39	90.57/91.27	74.71/76.71	84.80/85.19	85.03/85.76	79.80/82.21	80.01/79.81	83.69/84.24	81.34/82.32
7-	86.28/87.15	85.08/85.90	69.23/71.20	82.55/83.41	72.37/76.01	73.45/74.23	67.01/67.53	75.68/77.33	72.81/74.70

NOTATION: Each number is "Baseline-F1/Stag-F1," "root" indicates the root identification, and "1/2/3-6/7-" indicates the distance (the number of words) between a target word and its head.

+0.8 points), which suggests that it is better to encode the dependency label on the edge between the target word and its head for dependency parsing.

In addition to such head information, we wished to know which syntactic information of dependents could contribute to the improvements of UAS and LAS. To investigate this question, we compare the results of Stag-B/C/E, in which Stag-B encodes HLabel/Dir/hasDep, Stag-C encodes HLable/Dir/DLabel, and Stag-E encodes HLabel/Dir. Comparing Stag-C with Stag-E, they obtain much the same in UAS and LAS for both English and multilingual parsing settings. Comparing Stag-B with Stag-C, Stag-B outperformed Stag-C in both settings. These results suggest that the dependency labels of core arguments (DLabel) are not always effective and the dependent possession information (hasDep) contributes to the improvements of UAS and LAS.

Effects of Distance

To more deeply understand the characteristics of the parsers with supertags, we describe the parsing performance (F_1 score) according to the dependency distance, which represents the distance between a target word and its head word. Table IX shows the F1 scores of the baseline parser and parser with Stag-B (the supertag set that achieved the highest UAS/LAS on average) for each dependency distance.

Overall, the supertags helped improve the identification of the longer distance dependencies. For distances over seven words (7-), a performance boost is observed, yielding an increase of around 2.0 points on average over the six languages of UD. Similarly, English dependency parsing is improved by around 1.0 point.

The F1 scores of the root identification are improved on average as well. However, there is a performance gap between the languages. While the scores of English, German, Spanish, and Indonesian are drastically improved by the supertags, the scores of Arabic, Russian, and Chinese slightly decrease. A more detailed investigation of this is an interesting direction for future work.

TABLE X
UAS/LAS OF DEPENDENCY PARSERS IN PREVIOUS WORK

	UAS	LAS
Yamada & Matsumoto 2003 [10]	90.3	-
Zhang & Clark 2008 [14]	91.4	-
Huang & Sagae 2010 [19]	92.1	-
Zhang & Nivre 2011 [2]	92.9	91.8
Bohnet & Nivre 2012 [29]	**93.38**	92.44
Ouchi *et al.* 2014 [4]	91.35	-
Chen & Manning 2014 [30]	91.8	89.6
this work	92.94	91.67

C. Comparison With Existing Parsers

We compared our English parser with representative transition-based dependency parsing systems that use the Penn Treebank of Yamada & Matsumoto head rules (PTB-YM), i.e., the transition-based parser with a support vector machine of [10], pure transition-based parser of [14], dynamic-programming arc-standard parser of [19], arc-eager parser with rich non-local features of [2], transition-based joint POS tagging and parsing system of [29], easy-first parser with supertags of [4], and transition-based parser using neural networks of [30].

Table X shows the UAS and LAS on the test set. For our parser, we selected the highest scoring parser with STAG-B, and this parser is comparable to the previous systems. Although Ouchi *et al.* [4] used the same supertag set as STAG-B, our parser in this work outperformed it by over 1.5 points because we employed a more sophisticated parser, the arc-standard system with beam search.

However, the transition-based systems of [29] are slightly better than our parser. One of the possible explanations is that their system is a joint model for POS tagging and dependency parsing, and hence employs higher-order features, such as third-order features, which are not utilized in our system. They use such features by dynamically extracting them from the partial tree structures built during the parsing process (what we call dynamic features). Although such dynamic higher-order features are available after partial tree structures are constructed, they capture a wider context, which could lead to the high performance. On the contrary, supertags capture second-order information because they consist of head and dependent information, and supertag features are always available regardless of such partial tree structures, which help improve the parsing performance. As an interesting issue, it remains for us to determine how these different types of features interact with or complement each other when both features are leveraged in a transition-based system.

VIII. CONCLUSION

In this work, we presented a supertag design framework that is flexible so that various supertag sets may be designed. Based on the framework, we instantiated various granularity supertag sets that encode rich syntactic information. In previous work, syntactic information, such as the head and dependents of a word, cannot be used as features before partial tree structures are constructed [2]. However, by exploiting the supertags as features, we can utilize fine-grained syntactic information without waiting for partial trees to be built.

To investigate the utility of these supertag features, we have performed the experiments in multilingual dependency parsing as well as English parsing. The experimental results suggest the following:

1) Overall, our proposed supertag sets are effective for English and multilingual dependency parsing.
2) In particular, the supertag set that encodes the *head directionality/head labels/dependent possession* achieves the highest UAS and LAS.
3) Supertags contribute to the resolution of long distance dependencies.

Based on our proposed supertag design framework, we instantiated the seven supertag sets and used them as features for dependency parsers. For six languages that belong to different language branches as well as English, the supertag sets contributed to the improvements of UAS and LAS.

Comparing the results of the supertag sets, we found that in order to improve dependency parsing, it is critical to encode the head directionality, head label, and dependent possession information as supertags. In particular, the head label information is crucial for improving LAS. In contrast, the obligatory dependent labels do not improve the results.

Analyzing the results from the aspect of dependency distances, supertags especially contributed to the improvements in long distance dependency prediction. Long distance dependencies have been regarded as a troublesome problems in dependency parsing. Our experimental results suggest that supertags could be a solution to this problem.

As our future research, we would like to investigate the interaction of supertag features with higher-order features and explore linguistic entities that capture structurally richer information, such as subtree structures.

REFERENCES

[1] S. Kübler, R. McDonald, and J. Nivre, *Dependency Parsing*. San Rafael, CA, USA: Morgan and Clapool, 2009.

[2] Y. Zhang and J. Nivre, "Transition-based dependency parsing with rich non-local features," in *Proc. Annu. Meeting Assoc. Comput. Linguistics: Human Language Technol.*, 2011, pp. 188–193.

[3] T. Koo, X. Carreras, and M. Collins, "Simple semi-supervised dependency parsing," in *Proc. Annu. Meeting Assoc. Comput. Linguistics: Human Language Technol.*, 2008, pp. 595–603.

[4] H. Ouchi, K. Duh, and Y. Matsumoto, "Improving dependency parsers with supertags," in *Proc. Conf. Eur. Chapter Assoc. Comput. Linguistics*, 2014, pp. 154–158.

[5] A. Nasr and O. Rambow, "Supertagging and full parsing," in *Proc. 7th Int. Workshop Tree Adjoining Grammar Related Formalisms*, 2004, pp. 56–63.

[6] S. Bangalore and A. K. Joshi, "Supertagging: An approach to almost parsing," *Comput. Linguistics*, vol. 25, no. 2, pp. 237–265, 1999.

[7] K. Foth, T. By, and W. Menzel, "Guiding a constraint dependency parser with supertags," in *Proc. Int. Conf. Comput. Linguistics Annu. Meeting Assoc. Comput. Linguistics*, 2006, pp. 289–296.

[8] B. R. Ambati, T. Deoskar, and M. Steedman, "Improving dependency parsers using combinatory categorial grammar," in *Proc. Conf. Eur. Chapter Assoc. Comput. Linguistics*, 2014, pp. 159–163.

[9] T. Kudo and Y. Matsumoto, "Japanese dependency analysis using cascaded chunking," in *Proc. Conf. Nat. Lang. Learn.*, 2003, pp. 63–69.

[10] H. Yamada and Y. Matsumoto, "Statistical dependency analysis using support vector machines," in *Proc. Int. Workshop Parsing Technol.*, 2003, pp. 195–206.

[11] Y. Nivre, "An efficient algorithm for projective dependency parsing," in *Proc. Int. Workshop Parsing Technol.*, 2003, pp. 149–160.

[12] Y. Nivre, J. Hall, and J. Nilsson, "Memory-based dependency parsing," in *Proc. Conf. Nat. Language Learn.*, 2004, pp. 49–56.

[13] Y. Nivre and M. Scholz, "Deterministic dependency parsing of English text," in *Proc. Int. Conf. Comput. Linguistics*, 2004, pp. 64–70.

[14] Y. Zhang and S. Clark, "A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing using beam-search," in *Proc. Conf. Empirical Methods Natural Language Process.*, 2008, pp. 562–571.

[15] Y. Zhang and S. Clark, "Syntactic processing using the generalized perceptron and beam search," *Comput. Linguistics*, vol. 37, pp. 105–151, 2011.

[16] J. Nivre, "Algorithms for deterministic incremental dependency parsing," *Comput. Linguistics*, vol. 34, pp. 513–553, 2008.

[17] J. Nivre, "Incrementality in deterministic dependency parsing," in *Proc. Workshop Incremental Parsing: Bringing Eng. Cognition Together*, 2004, pp. 50–57.

[18] Y. Goldberg and M. Elhadad, "An efficient algorithm for easy-first non-directional dependency parsing," in *Proc. Human Language Technol.: Annu. Conf. North Amer. Chapter Assoc. Comput. Linguistics*, 2010, pp. 742–750.

[19] L. Huang and K. Sagae, "Dynamic programming for linear-time incremental parsing," in *Proc. Annu. Meeting Assoc. Comput. Linguistics*, 2010, pp. 1077–1086.

[20] M. P. Marcus, B. Santorini, and M. Marcinkiewicz, "Building a large annotated corpus of English: The Penn Treebank," *Comput. Linguistics*, vol. 19, no. 2, pp. 313–330, 1993.

[21] M.-C. de Marneffe, B. MacCartney, and C. D. Manning, "Generating typed dependency parses from phrase structure parses," in *Proc. Int. Conf. Language Resources Evaluation*, 2006, pp. 449–454.

[22] R. McDonald *et al.*, "Universal dependency annotation for multilingual parsing," in *Proc. Annu. Meeting Assoc. Comput. Linguistics*, 2013, pp. 92–97.

[23] M.-C. de Marneffe and C. D. Manning, "The Stanford typed dependencies representation," in *COLING-2008: Proc. Workshop Cross-Framework Cross-Domain Parser Evaluation*, 2008, pp. 1–8.

[24] M.-C. de Marneffe *et al.*, "Universal Stanford dependencies: A cross-linguistic typology," in *Proc. Int. Conf. Language Resources Evaluation*, 2014, pp. 4585–4592.

[25] S. Petrov, D. Das, and R. McDonald, "A universal part-of-speech tagset," in *Proc. Int. Conf. Language Resources Evaluation*, 2012, pp. 2089–2096.

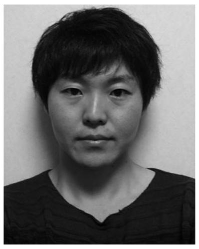
[26] D. Zeman, "A universal part-of-speech tagset," in *Proc. Int. Conf. Language Resources Evaluation*, 2008, pp. 213–218.

[27] M. Collins, "Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms," in *Proc. Conf. Empirical Methods Natural Language Process.*, 2002, pp. 1–8.

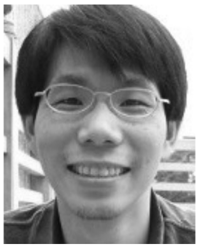
[28] L. Huang, S. Fayong, and Y. Guo, "Structured perceptron with inexact search," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Human Language Technol.*, 2012, pp. 142–151.

[29] B. Bohnet and J. Nivre, "A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing," in *Proc. Joint Conf. Empirical Methods Natural Language Process. Comput. Natural Language Learn.*, 2012, pp. 1455–1465.

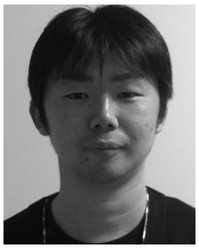
[30] D. Chen and C. Manning, "A fast and accurate dependency parser using neural networks," in *Proc. Conf. Empirical Methods Natural Language Process.*, 2014, pp. 740–750.



Hiroki Ouchi received the B.A. degree from Miyagi University of Education, Sendai, Japan, in 2011, the M.A. degree from Ritsumeikan University, Kyoto, Japan, in 2013, and the M.E. degree from Nara Institute of Science and Technology (NAIST), Ikoma, Japan in 2015. He is currently working toward the Ph.D. degree at the NAIST, Graduate School of Information Science. His research interests include natural language processing and machine learning, in particular, syntactic parsing and semantic role labeling.



Kevin Duh received the B.S. degree from Rice University, Houston, TX, USA, in 2003, and Ph.D. from the University of Washington, Seattle, WA, USA, in 2009, both in electrical engineering. He is currently working as an Assistant Professor at the Nara Institute of Science and Technology (NAIST), Graduate School of Information Science, Ikoma, Japan. Before joining NAIST, from 2009 to 2012, he worked at the NTT Communication Science Laboratories. His research interests include intersection of natural language processing and machine learning, in particular, in areas relating to machine translation and deep learning.



Hiroyuki Shindo received the B.E. and M.E. degrees from Waseda University, Japan, in 2007 and 2009, respectively, and the Ph.D. degree in engineering from Nara Institute of Science and Technology (NAIST), Ikoma, Japan, in 2013. He is currently working as an Assistant Professor at NAIST, Graduate School of Information Science. From 2009 to 2014, he was a Researcher at NTT Communication Science Laboratories. His research interests include machine learning and computational linguistics.

Yuji Matsumoto received the M.S. and Ph.D. degrees in information science from Kyoto University, Kyoto, Japan, in 1979 and 1989, respectively. He is currently working as a Professor of information science at the Nara Institute of Science and Technology, Ikoma, Japan. He joined the Machine Inference Section of Electrotechnical Laboratory in 1979. He has then gained experience as Academic Visitor at Imperial College of Science and Technology, a Deputy Chief of First Laboratory at ICOT, and an Associate Professor at Kyoto University. His research interests are natural language understanding and machine learning.