Integrating Multiple Dependency Corpora for Inducing Wide-coverage Japanese CCG Resources

Sumire Uematsu†
uematsu@cks.u-tokyo.ac.jp

Takuya Matsuzaki‡
takuya-matsuzaki@nii.ac.jp

Hiroki Hanaoka†
hanaoka@nii.ac.jp

Yusuke Miyao‡
yusuke@nii.ac.jp

Hideki Mima†
mima@t-adm.t.u-tokyo.ac.jp

†The University of Tokyo
Hongo 7-3-1, Bunkyo, Tokyo, Japan

‡National Institute of Infomatics
Hitotsubashi 2-1-2, Chiyoda, Tokyo, Japan

Abstract

This paper describes a method of inducing wide-coverage CCG resources for Japanese. While deep parsers with corpus-induced grammars have been emerging for some languages, those for Japanese have not been widely studied, mainly because most Japanese syntactic resources are dependency-based. Our method first integrates multiple dependency-based corpora into phrase structure trees and then converts the trees into CCG derivations. The method is empirically evaluated in terms of the coverage of the obtained lexicon and the accuracy of parsing.

1 Introduction

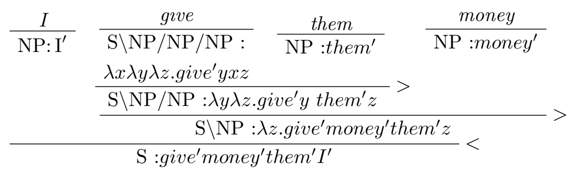
Syntactic parsing for Japanese has been dominated by a dependency-based pipeline in which chunk-based dependency parsing is applied and then semantic role labeling is performed on the dependencies (Sasano and Kurohashi, 2011; Kawahara and Kurohashi, 2011; Kudo and Matsumoto, 2002; Iida and Poesio, 2011; Hayashibe et al., 2011). This dominance is mainly because chunk-based dependency analysis looks most appropriate for Japanese syntax due to its morphosyntactic typology, which includes agglutination and scrambling (Bekki, 2010). However, it is also true that this type of analysis has prevented us from deeper syntactic analysis such as deep parsing (Clark and Curran, 2007) and logical inference (Bos et al., 2004; Bos, 2007), both of which have been surpassing shallow parsing-based approaches in languages like English.

In this paper, we present our work on inducing wide-coverage Japanese resources based on combinatory categorial grammar (CCG) (Steedman, 2001). Our work is basically an extension of a seminal work on CCGbank (Hockenmaier and Steedman, 2007), in which the phrase structure trees of the Penn Treebank (PTB) (Marcus et al., 1993) are converted into CCG derivations and a wide-coverage CCG lexicon is then extracted from these derivations. As CCGbank has enabled a variety of outstanding works on wide-coverage deep parsing for English, our resources are expected to significantly contribute to Japanese deep parsing.

The application of the CCGbank method to Japanese is not trivial, as resources like PTB are not available in Japanese. The widely used resources for parsing research are the Kyoto corpus (Kawahara et al., 2002) and the NAIST text corpus (Iida et al., 2007), both of which are based on the dependency structures of chunks. Moreover, the relation between chunk-based dependency structures and CCG derivations is not obvious.

In this work, we propose a method to integrate multiple dependency-based corpora into phrase structure trees augmented with predicate argument relations. We can then convert the phrase structure trees into CCG derivations. In the following, we describe the details of the integration method as well as Japanese-specific issues in the conversion into CCG derivations. The method is empirically evaluated in terms of the quality of the corpus conversion, the coverage of the obtained lexicon, and the accuracy of parsing with the obtained grammar. Additionally, we discuss problems that remain in Japanese resources from the viewpoint of developing CCG derivations.

There are three primary contributions of this paper: 1) we show the first comprehensive results for Japanese CCG parsing, 2) we present a methodology for integrating multiple dependency-based re-



I
NP: I′
give
S\NP/NP/NP : λxλyλz.give′yxz
them
NP :them′
money
NP :money′
>
S\NP/NP :λyλz.give′y them′z
>
S\NP :λz.give′money′them′z
<
S :give′money′them′I′

Figure 1: A CCG derivation.

$$
\begin{array}{llcll}
X/Y : f & Y : a & \to & X : fa & (>) \\
Y : a & X\backslash Y : a & \to & X : fa & (<) \\
X/Y : f & Y/Z : g & \to & X/Z : \lambda x.f(gx) & (> B) \\
Y\backslash Z : g & X\backslash Y : f & \to & X\backslash Z : \lambda x.f(gx) & (< B)
\end{array}
$$

Figure 2: Combinatory rules (used in the current implementation).

sources to induce CCG derivations, and 3) we investigate the possibility of further improving CCG analysis by additional resources.

2 Background

2.1 Combinatory Categorial Grammar

CCG is a syntactic theory widely accepted in the NLP field. A grammar based on CCG theory consists of *categories*, which represent syntactic categories of words and phrases, and *combinatory rules*, which are rules to combine the categories. Categories are either *ground categories* like S and NP or *complex categories* in the form of X/Y or $X\backslash Y$, where X and Y are the categories. Category X/Y intuitively means that it becomes category X when it is combined with another category Y to its right, and $X\backslash Y$ means it takes a category Y to its left. Categories are combined by applying combinatory rules (Fig. 2) to form categories for larger phrases. Figure 1 shows a CCG analysis of a simple English sentence, which is called a *derivation*. The verb *give* is assigned category $S\backslash NP/NP/NP$, which indicates that it takes two NPs to its right, one NP to its left, and finally becomes S. Starting from lexical categories assigned to words, we can obtain categories for phrases by applying the rules recursively.

An important property of CCG is a clear interface between syntax and semantics. As shown in Fig. 1, each category is associated with a lambda term of semantic representations, and each combinatory rule is associated with rules for semantic composition. Since these rules are universal, we can obtain different semantic representations by switching the semantic representations of lexical categories. This means that we can plug in a variety of semantic theories with CCG-based syntactic parsing (Bos et al., 2004).

Sentence	S	Verb	S\\$ (e.g. $S\backslash NP_{ga}$)
Noun phrase	NP	Post particle	$NP_{ga\|o\|ni\|to}\backslash NP$
Auxiliary verb	S\S		

Table 1: Typical categories for Japanese syntax.

Cat.	Feature	Value	Interpretation
NP	case	ga	nominal
		o	accusative
		ni	dative
		to	comitative, complementizer, etc.
		nc	none
S	form	stem	stem
		base	base
		neg	imperfect or negative
		cont	continuative
		vo_s	causative

Table 2: Features for Japanese syntax (those used in the examples in this paper).

2.2 CCG-based syntactic theory for Japanese

Bekki (2010) proposed a comprehensive theory for Japanese syntax based on CCG. While the theory is based on Steedman (2001), it provides concrete explanations for a variety of constructions of Japanese, such as agglutination, scrambling, long-distance dependencies, etc. (Fig. 3).

The ground categories in his theory are S, NP, and CONJ (for conjunctions). Table 1 presents typical lexical categories. While most of them are obvious from the theory of CCG, categories for auxiliary verbs require an explanation. In Japanese, auxiliary verbs are extensively used to express various semantic information, such as tense and modality. They agglutinate to the main verb in a sequential order. This is explained in Bekki's theory by the category S\S combined with a main verb via the function composition rule (<B). Syntactic features are assigned to categories NP and S (Table 2). The feature *case* represents a syntactic case of a noun phrase. The feature *form* denotes an inflection form, and is necessary for constraining the grammaticality of agglutination.

Our implementation of the grammar basically follows Bekki (2010)'s theory. However, as a first step in implementing a wide-coverage Japanese parser, we focused on the frequent syntactic constructions that are necessary for computing predicate argument relations, including agglutination, inflection, scrambling, case alternation, etc. Other details of the theory are largely simplified (Fig. 3),

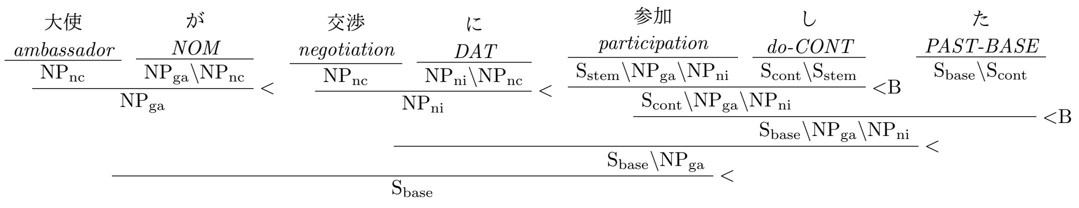


Figure 3: A simplified CCG analysis of the sentence "The ambassador participated in the negotiation.".

$S \rightarrow NP/NP$	(RelExt)
$S\backslash NP_1 \rightarrow NP_1/NP_1$	(RelIn)
$S \rightarrow S_1/S_1$	(Con)
$S\backslash\$_1\backslash NP_1 \rightarrow (S_1\backslash\$_1\backslash NP_1)/(S_1\backslash\$_1\backslash NP_1)$	(ConCoord)

Figure 4: Type changing rules. The upper two are for relative clauses and the others for continuous clauses.

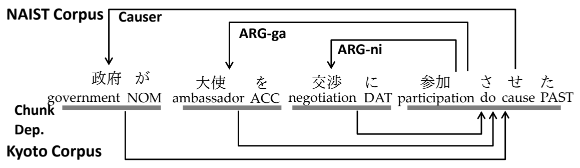


Figure 5: The Kyoto and NAIST annotations for "The government had the ambassador participate in the negotiation.". Accusatives are labeled as ARG-ga in causative (see Sec. 3.2).

coordination and semantic representation in particular. The current implementation recognizes coordinated verbs in continuous clauses (e.g., "彼はピアノを弾いて歌った/he played the piano and sang"), but the treatment of other types of coordination is largely simplified. For semantic representation, we define *predicate argument structures* (PASs) rather than the theory's formal representation based on dynamic logic. Sophisticating our semantic representation is left for future work.

For parsing efficiency, we modified the treatment of some constructions so that empty elements are excluded from the implementation. First, we define type changing rules to produce relative and continuous clauses (shown in Fig. 4). The rules produce almost the same results as the theory's treatment, but without using empty elements (*pro*, etc.). We also used lexical rules to treat pro-drop and scrambling. For the sentence in Fig. 3, the deletion of the nominal phrase (大使が), the dative phrase (交渉に), or both results in valid sentences, and shuffling the two phrases does so as well. Lexical entries with the scrambled or dropped arguments are produced by lexical rules in our implementation.

2.3 Linguistic resources for Japanese parsing

As described in Sec. 1, dependency-based analysis has been accepted for Japanese syntax. Research on Japanese parsing also relies on dependency-based corpora. Among them, we used the following resources in this work.

Kyoto corpus A news text corpus annotated with morphological information, chunk boundaries, and dependency relations among chunks (Fig. 5). The dependencies are classified into four types: Para (coordination), A (apposition), I (argument cluster), and Dep (default). Most of the dependencies are annotated as Dep.

NAIST text corpus A corpus annotated with anaphora and coreference relations. The same set as the Kyoto corpus is annotated.[1] The corpus only focuses on three cases: "ga" (subject), "o" (direct object), and "ni" (indirect object) (Fig. 5).

Japanese particle corpus (JP) (Hanaoka et al., 2010) A corpus annotated with distinct grammatical functions of the Japanese particle (postposition) "to". In Japanese, "to" has many functions, including a complementizer (similar to "that"), a subordinate conjunction (similar to "then"), a coordination conjunction (similar to "and"), and a case marker (similar to "with").

2.4 Related work

Research on Japanese deep parsing is fairly limited. Formal theories of Japanese syntax were presented by Gunji (1987) based on Head-driven Phrase Structure Grammar (HPSG) (Sag et al., 2003) and by Komagata (1999) based on CCG, although their implementations in real-world parsing have not been very successful. JACY (Siegel

[1]In fact, the NAIST text corpus includes additional texts, but in this work we only use the news text section.

and Bender, 2002) is a large-scale Japanese grammar based on HPSG, but its semantics is tightly embedded in the grammar and it is not as easy to systematically switch them as it is in CCG. Yoshida (2005) proposed methods for extracting a wide-coverage lexicon based on HPSG from a phrase structure treebank of Japanese. We largely extended their work by exploiting the standard chunk-based Japanese corpora and demonstrated the first results for Japanese deep parsing with grammar induced from large corpora.

Corpus-based acquisition of wide-coverage CCG resources has enjoyed great success for English (Hockenmaier and Steedman, 2007). In that method, PTB was converted into CCG-based derivations from which a wide-coverage CCG lexicon was extracted. CCGbank has been used for the development of wide-coverage CCG parsers (Clark and Curran, 2007). The same methodology has been applied to German (Hockenmaier, 2006), Italian (Bos et al., 2009), and Turkish (Çakıcı, 2005). Their treebanks are annotated with dependencies of *words*, the conversion of which into phrase structures is not a big concern. A notable contribution of the present work is a method for inducing CCG grammars from chunk-based dependency structures, which is not obvious, as we discuss later in this paper.

CCG parsing provides not only predicate argument relations but also CCG derivations, which can be used for various semantic processing tasks (Bos et al., 2004; Bos, 2007). Our work constitutes a starting point for such deep linguistic processing for languages like Japanese.

3 Corpus integration and conversion

For wide-coverage CCG parsing, we need a) a wide-coverage CCG lexicon, b) combinatory rules, c) training data for parse disambiguation, and d) a parser (e.g., a CKY parser). Since d) is grammar- and language-independent, all we have to develop for a new language is a)–c).

As we have adopted the method of CCGbank, which relies on a source treebank to be converted into CCG derivations, a critical issue to address is the absence of a Japanese counterpart to PTB. We only have chunk-based dependency corpora, and their relationship to CCG analysis is not clear.

Our solution is to first integrate multiple dependency-based resources and convert them into a phrase structure treebank that is independent

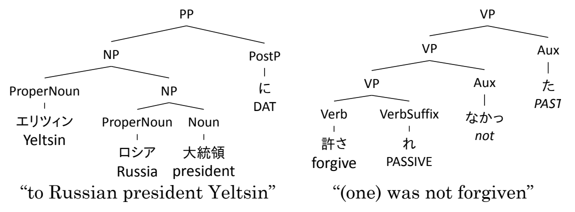


Figure 6: Internal structures of a nominal chunk (left) and a verbal chunk (right).

of CCG analysis (Step 1). Next, we translate the treebank into CCG derivations (Step 2). The idea of Step 2 is similar to what has been done with the English CCGbank, but obviously we have to address language-specific issues.

3.1 Dependencies to phrase structure trees

We first integrate and convert available Japanese corpora—namely, the Kyoto corpus, the NAIST text corpus, and the JP corpus —into a phrase structure treebank, which is similar in spirit to PTB. Our approach is to convert the dependency structures of the Kyoto corpus into phrase structures and then augment them with syntactic/semantic roles from the other two corpora.

The conversion involves two steps: 1) recognizing the chunk-internal structures, and (2) converting inter-chunk dependencies into phrase structures. For 1), we don't have any explicit information in the Kyoto corpus although, in principle, each chunk has internal structures (Vadas and Curran, 2007; Yamada et al., 2010). The lack of a chunk-internal structure makes the dependency-to-constituency conversion more complex than a similar procedure by Bos et al. (2009) that converts an Italian dependency treebank into constituency trees since their dependency trees are annotated down to the level of each word. For the current implementation, we abandon the idea of identifying exact structures and instead basically rely on the following generic rules (Fig. 6):

Nominal chunks Compound nouns are first formed as a right-branching phrase and post-positions are then attached to it.

Verbal chunks Verbal chunks are analyzed as left-branching structures.

The rules amount to assume that all but the last word in a compound noun modify the head noun (i.e., the last word) and that a verbal chunk is typically in a form $V\ A_1 \ldots A_n$, where V is a verb

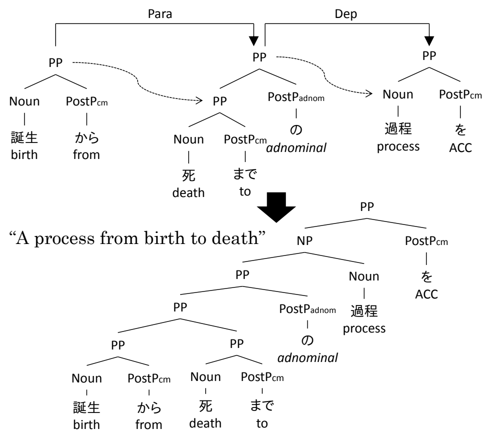


Figure 7: From inter-chunk dependencies to a tree.

(or other predicative word) and A_is are auxiliaries (see Fig. 6). We chose the left-branching structure as default for verbal chunks because the semantic scopes of the auxiliaries are generally in that order (i.e., A_1 has the narrowest scope). For both cases, phrase symbols are percolated upward from the right-most daughters of the branches (except for a few cases like punctuation) because in almost all cases the syntactic head of a Japanese phrase is the right-most element.

In practice, we have found several patterns of exceptions for the above rules. We implemented exceptional patterns as a small CFG and determined the chunk-internal structures by deterministic parsing with the generic rules and the CFG. For example, two of the rules we came up with are

rule A:	Number → PrefixOfNumber Number
rule B:	ClassifierPhrase → Number Classifier

in the precedence: rule A > B > generic rules. Using the above, we bracket a compound noun

約	千	人	死亡
approximately	*thousand*	*people*	*death*
PrefixOfNumber	Number	Classifier	CommonNoun

"*death of approximately one thousand people*"

as in

(((約	千)	人)	死亡)
(((approximately	thousand)	people)	death)

We can improve chunk-internal structures to some extent by refining the CFG rules. A complete solution like the manual annotation by Vadas and Curran (2007) is left for future work.

The conversion of inter-chunk dependencies into phrase structures may sound trivial, but it is not necessarily easy when combined with chunk-internal structures. The problem is to which node in the internal structure of the head the dependent tree is adjoined (Fig. 7). In the case shown in the figure, three chunks are in the dependency relation indicated by arrows on the top. The dotted arrows show the nodes to which the subtrees are adjoined.

dep	modifier-type	precedence
Para	から/PostP$_{\mathrm{cm}}$	まで/PostP$_{\mathrm{cm}}$, */(Verb|Aux), ...
Dep	*/PostP$_{\mathrm{cm}}$	*/(Verb|Aux), */Noun, ...
Dep	*/PostP$_{\mathrm{adnom}}$	*/Noun, */(Verb|Aux), ...

Table 3: Rules to determine adjoin position.

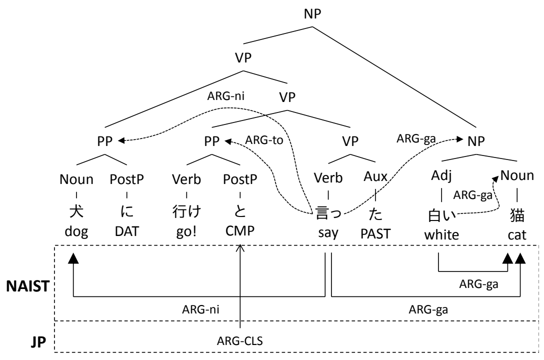


Figure 8: Overlay of pred-arg structure annotation ("The white cat who said "Go!" to the dog.").

Without any human-created resources, we cannot always determine the adjoin positions correctly. Therefore, as a compromise, we wound up implementing approximate heuristic rules to determine the adjoin positions. Table 3 shows examples of such rules. A rule specifies a precedence of the possible adjoin nodes as an ordered list of patterns on the lexical head of the subtree under an adjoin position. The precedence is defined for each combination of the type of the dependent phrase, which is determined by its lexical head, and the dependency type in the Kyoto corpus.

To select the adjoin position for the left-most subtree in Fig. 7, for instance, we look up the rule table using the dependency type, "Para", and the lexical head of the modifier subtree, "から/PostP$_{\mathrm{cm}}$", as the key, and find the precedence "まで/PostP$_{\mathrm{cm}}$, */(Verb|Aux), ...". We thus select the PP-node on the middle subtree indicated by the dotted arrow because its lexical head (the right-most word), "まで/PostP$_{\mathrm{cm}}$", matches the first pattern in the precedence list. In general, we seek for an adjoin node for each pattern p in the precedence list, until we find a first match.

The semantic annotation given in the NAIST corpus and the JP corpus is overlaid on the phrase structure trees with slight modifications (Fig. 8).

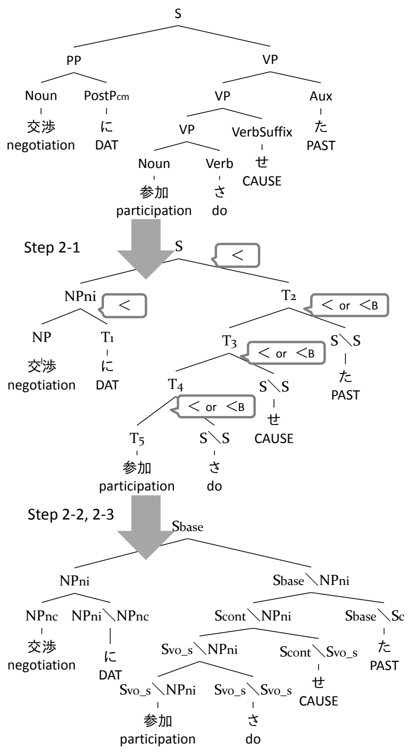


Figure 9: A phrase structure into a CCG derivation.

In the figure, the annotation given in the two corpora is shown inside the dotted box at the bottom. We converted the predicate-argument annotations given as labeled word-to-word dependencies into the relations between the predicate words and their argument *phrases*. The results are thus similar to the annotation style of PropBank (Palmer et al., 2005). In the NAIST corpus, each pred-arg relation is labeled with the argument-type (ga/o/ni) and a flag indicating that the relation is mediated by either a syntactic dependency or a zero anaphora. For a relation of a predicate w_p and its argument w_a in the NAIST corpus, the boundary of the argument phrase is determined as follows:

1. If w_a precedes w_p and the relation is mediated by a syntactic dep., select the maximum PP that is formed by attaching one or more postpositions to the NP headed by w_a.
2. If w_p precedes w_a or the relation is mediated by a zero anaphora, select the maximum NP headed by w_a that does not include w_p.

In the figure, "犬/dog に/DAT" is marked as the ni-argument of the predicate "言っ/say" (Case 1), and "白い/white 猫/cat" is marked as its ga-argument (Case 2). Case 1 is for the most basic construction, where an argument PP precedes its predicate. Case 2 covers the relative clause construction, where a relative clause precedes the head NP, the modification of a noun by an adjective, and the relations mediated by zero anaphora.

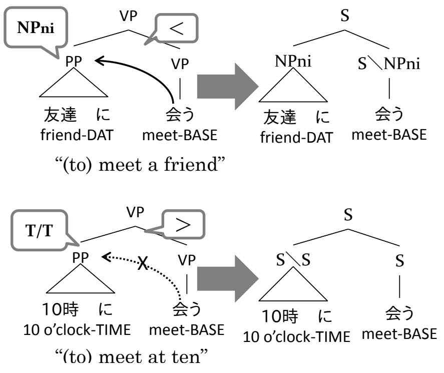


Figure 10: An argument post particle phrase (PP) (upper) and an adjunct PP (lower).

The JP corpus provides only the function label to each particle "to" in the text. We determined the argument phrases marked by the "to" particles labeled as (nominal or clausal) argument-markers in a similar way to Case 1 above and identified the predicate words as the lexical heads of the phrases to which the PP_{to} phrases attach.

3.2 Phrase structures to CCG derivations

This step consists of three procedures (Fig. 9):

1. Add constraints on categories and features to tree nodes as far as possible and assign a combinatory rule to each branching.
2. Apply combinatory rules to all branching and obtain CCG derivations.
3. Add feature constraints to terminal nodes.

3.2.1 Local constraint on derivations

According to the phrase structures, the first procedure in Step 2 imposes restrictions on the resulting CCG derivations. To describe the restrictions, we focus on some of the notable constructions and illustrate the restrictions for each of them.

Phrases headed by case marker particles A phrase of this type must be either an argument (Fig. 10, upper) or a modifier (Fig. 10, lower) of a predicative. Distinction between the two is made based on the pred-arg annotation of the predicative. If a phrase is found to be an argument, 1) category NP is assigned to the corresponding node, 2) the case feature of the category is given according to the particle (in the case of Fig. 10 (upper),

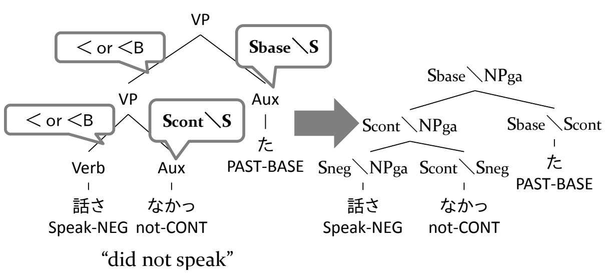


Figure 11: An auxiliary verb and its conversion.

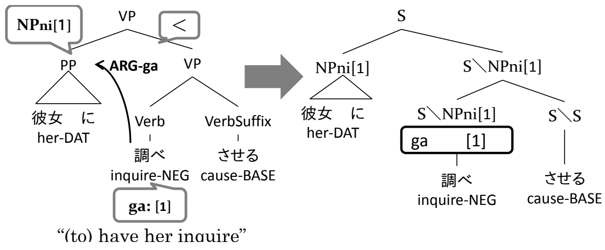


Figure 12: A causative construction.

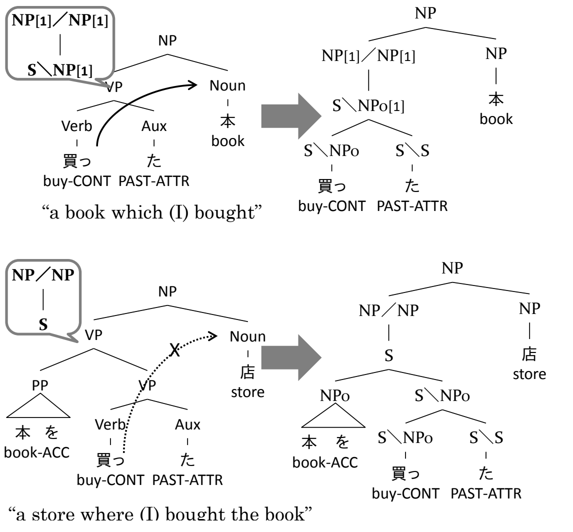


Figure 13: A relative clause with/without argument extraction (upper/lower, respectively).

ni for dative), and 3) the combinatory rule that combines the particle phrase and the predicative phrase is assigned *backward function application rule* (<). Otherwise, a category T/T is assigned to the corresponding modifier node and the rule will be *forward function application* (>).

Auxiliary verbs As described in Sec. 2.2, an auxiliary verb is always given the category $S\backslash S$ and is combined with a verbal phrase via < or <B (Fig. 11). Furthermore, we assign the *form* feature value of the returning category S according to the inflection form of the auxiliary. In the case shown in the figure, $S_{base}\backslash S$ is assigned for "た/PAST-BASE" and $S_{cont}\backslash S$ for "なかっ/not-CONT". As a result of this restriction, we can obtain conditions for every auxiliary agglutination because the two *form* values in $S\backslash S$ are both restricted after applying combinatory rules (Sec. 3.2.2).

Case alternations In addition to the argument/adjunct distinction illustrated above, a process is needed for argument phrases of predicates involving case alternation. Such predicates are either causative (see Fig. 12) or passive verbs and can be detected by voice annotation from the NAIST corpus. For an argument of that type of verb, its *deep* case (*ga* for Fig. 12) must be used to construct the semantic representation, namely the PAS. As well as assigning the shallow case value (*ni* in Fig. 12) to the argument's category NP, as usual, we assign a restriction to the PAS of the verb so that the semantic argument corresponding to the deep case is co-indexed with the argument NP. These restrictions are then utilized for PAS construction in Sec. 3.2.3.

Relative clauses A relative clause can be detected as a subtree that has a VP as its left child and an NP as its right child, as shown in Fig. 13. The conversion of the subtree consists of 1) inserting a node on the top of the left VP (see the right-hand side of Fig. 13), and 2) assigning the appropriate unary rule to make the new node. The difference between candidate rules RelExt and RelIn (see Fig. 4) is whether the right-hand NP is an obligatory argument of the VP or not, which can be determined by the pred-arg annotation on the predicate in the VP. In the upper example in Fig. 13, RelIn is assigned because the right NP "book" is annotated as an accusative argument of the predicate "buy". In contrast, RelExt is assigned in the lower side in the figure because the right NP "store" is not annotated as an argument.

Continuous clauses A continuous clause can be detected as a subtree with a VP of continuous form as its left child and a VP as its right child. Its conversion is similar to that of a relative clause, and only differs in that the candidate rules are Con and ConCoord. ConCoord generates a continuous clause that shares arguments with the main clause while Con produces one without shared arguments. Rule assignment is done by comparing the pred-arg annotations of the two phrases.

	Training	Develop.	Test
#Sentences	24,283	4,833	9,284
#Chunks	234,685	47,571	89,874
#Words	664,898	136,585	255,624

Table 4: Statistics of input linguistic resources.

3.2.2 Inverse application of rules

The second procedure in Step 2 begins with assigning a category S to the root node. A combinatory rule assigned to each branching is then "inversely" applied so that the constraint assigned to the parent transfers to the children.

3.2.3 Constraints on terminal nodes

The final process consists of a) imposing restrictions on the terminal category in order to instantiate all the feature values, and b) constructing a PAS for each verbal terminal. An example of process a) includes setting the *form* features in the verb category, such as $\mathrm{S}\backslash\mathrm{NP}_{ni}$, according to the inflection form of the verb. As for b), arguments in a PAS are given according to the category and the partial restriction. For instance, if a category $\mathrm{S}\backslash\mathrm{NP}_{ni}$ is obtained for "調べ/inquire" (Fig. 12), the PAS for "inquire" is unary because the category has one argument category (NP_{ni}), and the category is co-indexed with the semantic argument *ga* in the PAS due to the partial restriction depicted in Sec. 3.2.1. As a result, a lexical entry is obtained as 調べ $\vdash \mathrm{S}\backslash\mathrm{NP}_{ni}[1]$: inquire([1]).

3.3 Lexical entries

Finally, lexical rules are applied to each of the obtained lexical entries in order to reduce them to the canonical form. Since words in the corpus (especially verbs) often involve pro-drop and scrambling, there are a lot of obtained entries that have slightly varied categories yet share a PAS. We assume that an obtained entry is a variation of the canonical one and register the canonical entries in the lexicon. We treat only subject deletion for pro-drop because there is not sufficient information to judge the deletion of other arguments. Scrambling is simply treated as permutation of arguments.

4 Evaluation

We used the following for the implementation of our resources: Kyoto corpus ver. 4.0[2], NAIST text corpus ver. 1.5[3], and JP corpus ver. 1.0[4]. The integrated corpus is divided into training, development, and final test sets following the standard data split in previous works on Japanese dependency parsing (Kudo and Matsumoto, 2002). The details of these resources are shown in Table 4.

	Training		Develop.		Test	
	St.1	St.2	St.1	St.2	St.1	St.2
Sent.	24,283	24,116	4,833	4,803	9,284	9,245
Converted	24,116	22,820	4,803	4,559	9,245	8,769
Con. rate	99.3	94.6	99.4	94.9	99.6	94.9

Table 5: Statistics of corpus conversion.

Sentential Coverage

	Covered	Uncovered	Cov. (%)
Devel.	3,920	639	85.99
Test	7,610	1,159	86.78

Lexical Coverage

	Word	Known	Unknown		
			combi.	cat.	word
Devel.	127,144	126,383	682	79	0
Test	238,083	236,651	1,242	145	0

Table 6: Sentential and lexical coverage.

4.1 Corpus conversion and lexicon extraction

Table 5 shows the number of successful conversions performed by our method. In total, we obtained 22,820 CCG derivations from 24,283 sentences (in the training set), resulting in the total conversion rate of 93.98%. The table shows we lost more sentences in Step 2 than in Step 1. This is natural because Step 2 imposed more restrictions on resulting structures and therefore detected more discrepancies including compounding errors. Our conversion rate is about 5.5 points lower than the English counterpart (Hockenmaier and Steedman, 2007). Manual investigation of the sampled derivations would be beneficial for the conversion improvement.

For the lexicon extraction from the CCGbank, we obtained 699 types of lexical categories from 616,305 word tokens. After lexical reduction, the number of categories decreased to 454, which in turn may produce 5,342 categories by lexical expansion. The average number of categories for a word type was 11.68 as a result.

4.2 Evaluation of coverage

Following the evaluation criteria in (Hockenmaier and Steedman, 2007), we measured the coverage

[2] http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?Kyoto\%20University\%20Text\%20Corpus

[3] http://cl.naist.jp/nldata/corpus/

[4] https://alaginrc.nict.go.jp/resources/tocorpus/tocorpusabstract.html

of the grammar on unseen texts. First, we obtained CCG derivations for evaluation sets by applying our conversion method and then used these derivations as gold standard. Lexical coverage indicates the number of words to which the grammar assigns a gold standard category. Sentential coverage indicates the number of sentences in which all words are assigned gold standard categories [5].

Table 6 shows the evaluation results. Lexical coverage was 99.40% with rare word treatment, which is in the same level as the case of the English CCG parser C&C (Clark and Curran, 2007). We also measured coverage in a "weak" sense, which means the number of sentences that are given at least one analysis (not necessarily correct) by the obtained grammar. This number was 99.12 % and 99.06 % for the development and the test set, respectively, which is sufficiently high for wide-coverage parsing of real-world texts.

4.3 Evaluation of parsing accuracy

Finally, we evaluated the parsing accuracy. We employed the parser and the supertagger of (Miyao and Tsujii, 2008), specifically, its generalized modules for lexicalized grammars. We trained log-linear models in the same way as (Clark and Curran, 2007) using the training set as training data. Feature sets were simply borrowed from an English parser; no tuning was performed. Following conventions in research on Japanese dependency parsing, gold morphological analysis results were input to a parser. Following C&C, the evaluation measure was precision and recall over dependencies, where a dependency is defined as a 4-tuple: a head of a functor, a functor category, an argument slot, and a head of an argument.

Table 7 shows the parsing accuracy on the development and the test sets. The supertagging accuracy is presented in the upper table. While our coverage was almost the same as C&C, the performance of our supertagger and parser was lower. To improve the performance, tuning disambiguation models for Japanese is a possible approach. Comparing the parser's performance with previous works on Japanese dependency parsing is difficult as our figures are not directly comparable to theirs. Sassano and Kurohashi (2009) reported the accuracy of their parser as 88.48 and 95.09 in unlabeled chunk-based and word-based F1 respectively. While our score of 90.05 in unlabeled category dependency seems to be lower than their word-based score, this is reasonable because our category dependency includes more difficult problems, such as whether a subject PP is shared by coordinated verbs. Thus, our parser is expected to be capable of real-world Japanese text analysis as well as dependency parsers.

[5]Since a gold derivation can logically be obtained if gold categories are assigned to all words in a sentence, sentential coverage means that the obtained lexicon has the ability to produce exactly correct derivations for those sentences.

Supertagging accuracy

	Lex. Cov.	Cat. Acc.
Devel.	99.40	90.86
Test	99.40	90.69
C&C	99.63	94.32

Overall performance

	LP	LR	LF	UP	UR	UF
Devel.	82.55	82.73	82.64	90.02	90.22	90.12
Test	82.40	82.59	82.50	89.95	90.15	90.05
C&C	88.34	86.96	87.64	93.74	92.28	93.00

Table 7: Parsing accuracy. LP, LR and LF refer to labeled precision, recall, and F-score respectively. UP, UR, and UF are for unlabeled.

5 Conclusion

In this paper, we proposed a method to induce wide-coverage Japanese resources based on CCG that will lead to deeper syntactic analysis for Japanese and presented empirical evaluation in terms of the quality of the obtained lexicon and the parsing accuracy. Although our work is basically in line with CCGbank, the application of the method to Japanese is not trivial due to the fact that the relationship between chunk-based dependency structures and CCG derivations is not obvious.

Our method integrates multiple dependency-based resources to convert them into an integrated phrase structure treebank. The obtained treebank is then transformed into CCG derivations. The empirical evaluation in Sec. 4 shows that our corpus conversion successfully converts 94 % of the corpus sentences and the coverage of the lexicon is 99.4 %, which is sufficiently high for analyzing real-world texts. A comparison of the parsing accuracy with previous works on Japanese dependency parsing and English CCG parsing indicates that our parser can analyze real-world Japanese texts fairly well and that there is room for improvement in disambiguation models.

References

Daisuke Bekki. 2010. *Formal Theory of Japanese Syntax*. Kuroshio Shuppan. (In Japanese).

Johan Bos, Stephen Clark, Mark Steedman, James R. Curran, and Julia Hockenmaier. 2004. Wide-coverage semantic representations from a CCG parser. In *Proceedings of COLING 2004*, pages 1240–1246.

Johan Bos, Cristina Bosco, and Alessandro Mazzei. 2009. Converting a dependency treebank to a categorial grammar treebank for Italian. In *Proceedings of the Eighth International Workshop on Treebanks and Linguistic Theories (TLT8)*, pages 27–38.

Johan Bos. 2007. Recognising textual entailment and computational semantics. In *Proceedings of Seventh International Workshop on Computational Semantics IWCS-7*, page 1.

Ruken Çakıcı. 2005. Automatic induction of a CCG grammar for Turkish. In *Proceedings of ACL Student Research Workshop*, pages 73–78.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4).

Takao Gunji. 1987. *Japanese Phrase Structure Grammar: A Unification-based Approach*. D. Reidel.

Hiroki Hanaoka, Hideki Mima, and Jun'ichi Tsujii. 2010. A Japanese particle corpus built by example-based annotation. In *Proceedings of LREC 2010*.

Yuta Hayashibe, Mamoru Komachi, and Yuji Matsumoto. 2011. Japanese predicate argument structure analysis exploiting argument position and type. In *Proceedings of IJCNLP 2011*, pages 201–209.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Julia Hockenmaier. 2006. Creating a CCGbank and a wide-coverage CCG lexicon for German. In *Proceedings of the Joint Conference of COLING/ACL 2006*.

Ryu Iida and Massimo Poesio. 2011. A cross-lingual ILP solution to zero anaphora resolution. In *Proceedings of ACL-HLT 2011*, pages 804–813.

Ryu Iida, Mamoru Komachi, Kentaro Inui, and Yuji Matsumoto. 2007. Annotating a Japanese text corpus with predicate-argument and coreference relations. In *Proceedings of Linguistic Annotation Workshop*, pages 132–139.

Daisuke Kawahara and Sadao Kurohashi. 2011. Generative modeling of coordination by factoring parallelism and selectional preferences. In *Proceedings of IJCNLP 2011*.

Daisuke Kawahara, Sadao Kurohashi, and Koiti Hasida. 2002. Construction of a Japanese relevance-tagged corpus. In *Proceedings of the 8th Annual Meeting of the Association for Natural Language Processing*, pages 495–498. (In Japanese).

Nobo Komagata. 1999. *Information Structure in Texts: A Computational Analysis of Contextual Appropriateness in English and Japanese*. Ph.D. thesis, University of Pennsylvania.

Taku Kudo and Yuji Matsumoto. 2002. Japanese dependency analyisis using cascaded chunking. In *Proceedings of CoNLL 2002*.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Yusuke Miyao and Jun'ichi Tsujii. 2008. Feature forest models for probabilistic HPSG parsing. *Computational Linguistics*, 34(1):35–80.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Ivan A. Sag, Thomas Wasow, and Emily M. Bender. 2003. *Syntactic Theory: A Formal Introduction, 2nd Edition*. CSLI Publications.

Ryohei Sasano and Sadao Kurohashi. 2011. A discriminative approach to Japanese zero anaphora resolution with large-scale lexicalized case frames. In *Proceedings of IJCNLP 2011*.

Manabu Sassano and Sadao Kurohashi. 2009. A unified single scan algorithm for Japanese base phrase chunking and dependency parsing. In *Proceedings of ACL-IJCNLP 2009*.

Melanie Siegel and Emily M. Bender. 2002. Efficient deep processing of Japanese. In *Proceedings of the 3rd Workshop on Asian Language Resources and International Standardization*.

Mark Steedman. 2001. *The Syntactic Process*. MIT Press.

David Vadas and James Curran. 2007. Adding noun phrase structure to the Penn Treebank. In *Proceedings of ACL 2007*, pages 240–247.

Emiko Yamada, Eiji Aramaki, Takeshi Imai, and Kazuhiko Ohe. 2010. Internal structure of a disease name and its application for ICD coding. *Studies in health technology and informatics*, 160(2):1010–1014.

Kazuhiro Yoshida. 2005. Corpus-oriented development of Japanese HPSG parsers. In *Proceedings of the ACL Student Research Workshop*.