Reforging : A Method for Constructing a Linguistically Valid Japanese CCG Treebank

Asa Tomita [1] **Hitomi Yanaka** [2] **Daisuke Bekki** [1]
[1] Ochanomizu University, Japan
[2] The University of Tokyo, Japan
{tomita.asa,bekki}@is.ocha.ac.jp
hyanaka@is.s.u-tokyo.ac.jp

Abstract

The linguistic validity of Combinatory Categorial Grammar (CCG) parsing results relies heavily on treebanks for training and evaluation, so the treebank construction is crucial. Yet the current Japanese CCG treebank is known to have inaccuracies in its analyses of Japanese syntactic structures, including passive and causative constructions. While ABCTreebank, a treebank for ABC grammar, has been made to improve the analysis, particularly of argument structures, it lacks the detailed syntactic features required for Japanese CCG. In contrast, the Japanese CCG parser, *lightblue*, efficiently provides detailed syntactic features, but it does not accurately capture argument structures. We propose a method to generate a linguistically valid Japanese CCG treebank with detailed information by combining the strengths of ABCTreebank and *lightblue*. We develop an algorithm that filters *lightblue*'s lexical items using ABCTreebank, effectively converting *lightblue* output into a linguistically valid CCG treebank. To evaluate our treebank, we manually evaluate CCG syntactic structures and semantic representations and analyze conversion rates.

1 Introduction

There have been significant advances in natural language processing research through the construction of syntactic tree corpora, known as treebanks. Treebanks are datasets where syntactic structures are annotated over large bodies of text. Various treebanks (Marcus et al., 1993; Forst, 2003; Briscoe and Carroll, 2006; Hockenmaier, 2006; Hockenmaier and Steedman, 2007; Vadas and Curran, 2007; Bos et al., 2010; Boxwell and Brew, 2010) have been served as standard datasets for training and evaluating statistical syntactic parsers. The Penn Treebank (Marcus et al., 1993), one of the first context-free grammar (CFG) treebanks, contains a one-million-word corpus of *Wall Street Journal* text. CCGbank (Hockenmaier and Steedman, 2007) was constructed by converting the Penn Treebank to Combinatory Categorial Grammar (CCG; Steedman, 1996, 2000), which contributing to the advancement of CCG parsers.

There are various methods for constructing treebanks. One approach is to combine automatic part-of-speech (POS) taggers and syntactic parsers with manual corrections, as in the Penn Treebank. The approach to providing CCGbank involves automatic conversion from existing treebanks. However, treebanks have different formats, provided information, and informational validity. Japanese CCGbank (Uematsu et al., 2013), constructed by automatic conversion of Japansese dependency tree corpora, but provides limited validity of syntactic structures for passive or causative nestings (Bekki and Yanaka, 2023). The Japanese ABC grammar treebank ABCTreebank (Kubota et al., 2020) has manually annotated argument structures, but does not provide POS information (conjugation series, conjugation forms, among others) and other detailed information.

We thus aim to construct a Japanese CCG treebank with both linguistically valid syntactic structures and detailed syntactic features. To this end, we propose a method to construct a new Japanese CCG treebank using the Japanese CCG parser *lightblue* (Bekki and Kawazoe, 2016)[1], which can output detailed syntactic features. However, *lightblue* contains errors related to argument structures, causing inaccurate outputs. To address this drawback, we extracted predicate-argument structures from ABCTreebank and incorporated this information into *lightblue*. This approach involves decomposing and reconstructing the treebank, which we call "reforging". We discuss the reforging process in more detail in Section 3. Section 4 assess the validity of our proposed method. Section 5 introduces the error analysis of the output trees. Section 6

[1] https://github.com/DaisukeBekki/lightblue

describes manual evaluations considering the syntactic structures and semantic representations of the output trees.

2 Background

2.1 Combinatory Categorial Grammar

CCG is a lexicalized grammar consisting of combinatory rules and a lexicon. Syntactic categories are either base categories or functional categories. The set of base categories includes elements such as NP for *noun phrases* and S for *sentences*. Functional categories use slash notation to represent complex phrases. For instance, the functional categories can express an intransitive verb as $S\backslash NP$, finding the syntactic structure of category NP on the right and returning S. Slash and backslash notations as $(S\backslash NP)/NP$ express transitive verbs.

In CCG, lexicons associate words with their phonological and syntactic information. For instance, to analyze the sentence

(1) Taro runs.

in CCG, the the lexical items

$$\begin{array}{ll} Taro: & NP \\ runs: & S\backslash NP \end{array}$$

are supposed to be contained in the lexicon. Combinatory rules allow syntactic categories to be merged. Function application rules and function composition rules are the basic CCG rules, defined as follows:

1. Function application rule
   (a) Forward application (>)
   $X/Y \quad Y \quad \Rightarrow \quad X$
   (b) Backward application (<)
   $Y \quad X\backslash Y \quad \Rightarrow \quad X$
2. Function composition rule
   (a) Forward composition (>B)
   $X/Y \quad Y/Z \quad \Rightarrow \quad X/Z$
   (b) Backward composition (<B)
   $Y\backslash Z \quad X\backslash Y \quad \Rightarrow \quad X\backslash Z$

The CCG syntactic structure of the sentence (1) is given using the following function application rule:

$$\dfrac{\dfrac{\text{Taro}}{NP} \quad \dfrac{\text{runs}}{S\backslash NP}}{S} <$$

In CCG, a combinatory rule applied to syntactic structures is indicated by a symbol placed on the right of the horizontal line. For example, in the above syntactic structure, the symbol "<" on the right of the horizontal line indicates the use of the backward application rule. CCG also includes other rules such as coordination, crossed-composition, and crossed-substitution rules.

2.2 Japanese CCGbank

Uematsu et al. (2013) constructed Japanese CCGbank through automatic conversion of the Kyoto corpus[2], NAIST text corpus[3], and Japanese particle corpus (Hanaoka et al., 2010). However, Japanese CCGbank has some empirical problems. One of the problems, discussed by Bekki and Yanaka (Bekki and Yanaka, 2023), is that the syntactic analysis of the Japanese CCGbank contains empirical fallacies on predictions for passive and causative nestings. For instance, consider the passive sentence:

(2) 太郎が 次郎に 褒めら れ た
Taro-ga Jiro-ni homera re ta
Taro-NOM Jiro-DAT praise passive PST
'Taro was praised by Jiro.'

Figure 1 shows a syntactic structure based on Japanese CCGBank, which assigns the category $S\backslash S$ to the passive suffix *re*. However, *re* needs to play a role in changing the argument structure of the transitive verb *homera*, so this analysis is invalid.

Figure 2 shows a syntactic structure that can be analyzed based on Japanese CCG (Bekki, 2010). In this syntactic structure, the syntactic category $S\backslash NP_{ga}\backslash NP_{ni}\backslash(S\backslash NP_{ga}\backslash NP_{ni|o})$ is assigned to the passive suffix *re* and $S\backslash NP_{ga}\backslash NP_o$ to the predicate *homera*. By combining *homera* with *re* using the function composition rule, the argument structure of the predicate *homera* is changed to $S\backslash NP_{ga}\backslash NP_{ni}$, so it takes nominative and dative noun phrases as arguments. Thus, this syntactic structure indicates a valid passive nesting.

2.3 ABCTreebank

ABCTreebank was constructed in an attempt to create a general-purpose treebank. It was constructed by converting the Keyaki Treebank[4], a phrase-structured treebank, to ABC grammar trees. The

[2] https://github.com/ku-nlp/KyotoCorpus
[3] https://sites.google.com/site/naisttextcorpus/
[4] https://github.com/ajb129/KeyakiTreebank

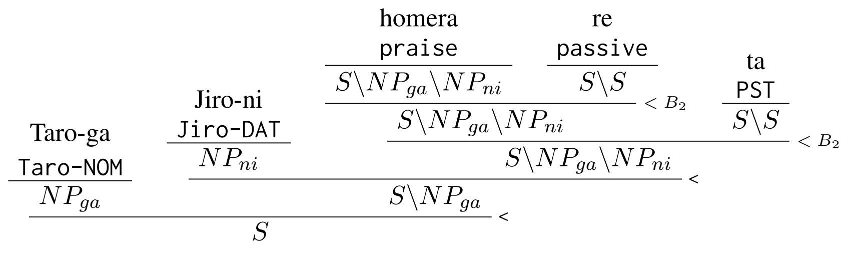
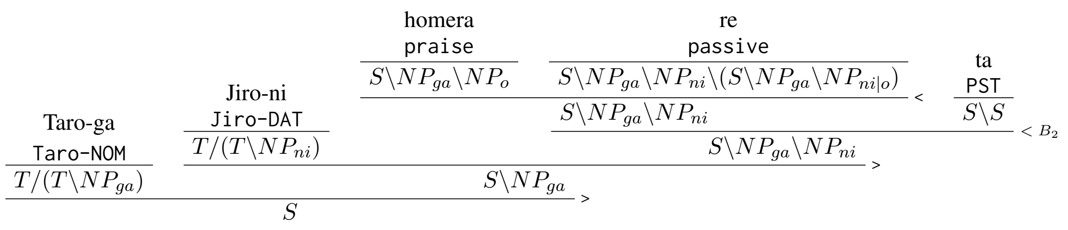


Figure 1: A syntactic structure based on Japanese CCGbank



Taro-ga
Taro-NOM
$T/(T\backslash NP_{ga})$
Jiro-ni
Jiro-DAT
$T/(T\backslash NP_{ni})$
homera
praise
$S\backslash NP_{ga}\backslash NP_{o}$
re
passive
$S\backslash NP_{ga}\backslash NP_{ni}\backslash(S\backslash NP_{ga}\backslash NP_{ni|o})$
$<$
$S\backslash NP_{ga}\backslash NP_{ni}$
ta
PST
$S\backslash S$
$< B_2$
$S\backslash NP_{ga}\backslash NP_{ni}$
$>$
$S\backslash NP_{ga}$
$>$
S

Figure 2: A syntactic structure based on Bekki (2010)

ABC grammar consists of two CCG rules: function application and function composition rules. Since these basic rules are common in both CCG and type-logical grammar (TLG; Morrill, 1994; Moortgat, 1997), syntactic structures in ABCTreebank can be easily converted to CCG syntactic structures.

ABCTreebank argument structures are assumed to be reliable because they were manually annotated. However, ABCTreebank does not cover the syntactic information, such as POS information. Incorporating POS information into ABCTreebank's syntactic structure is challenging because CCG syntactic structures cannot be retrospectively recovered and have more elaborate information than ABC syntactic structures, such as syntactic features. Syntactic features can contain diverse information, such as person agreement, number, gender, tense, and case frame. For instance, the lexical item of the verb *runs* in sentence (1) is written using syntactic features as follows.

$$runs: \qquad S\backslash NP_{3S}$$

The syntactic category NP_{3S} denotes a third-person singular noun phrase. The syntactic structure of the sentence (1) then appears as follows:

$$\dfrac{\dfrac{\text{Taro}}{NP_{3SM}} \qquad \dfrac{\text{runs}}{S\backslash NP_{3S}}}{S}<$$

The syntactic category NP_{3SM} is a third-person singular noun phrase indicating that Taro is male.

Although morphological analyzers, such as Juman (Kawahara and Kurohashi, 2006), can generate elaborate syntactic structures, mapping this information to ABCTreebank's syntactic structure is challenging, as the elaborate syntactic information needs to be supplemented with less informative syntactic structure. Machine learning approaches for such mappings require annotated training data. However, no annotated data currently exists, so there is no method for recovering CCG syntactic structures. Consequently, the treebank itself must possess elaborate information.

2.4 *lightblue*

lightblue is a Japanese CCG parser based on Bekki (2010) that outputs CCG syntactic structures with detailed syntactic features. Note that *lightblue* computes syntactic structures from lexicon and combinatory rules, so unlike other parsers, it does not require training and evaluation data. Its lexicon contains about 80,000 words with case frames extracted from the Juman dictionary. *lightblue* also provides semantic representations in terms of Dependent Type Semantics (DTS; Bekki and Mineshima, 2017) as shown in Figure 4. The phonetic form appears above the tree's horizontal line, and the CCG syntactic category and the DTS semantic representation are displayed below the horizontal line. The symbol to the right is the applicable CCG combinatory rule or the identifier of the lexicon in which the word is registered. The current version of *lightblue*'s argument structures include some errors, leading to unnatural disambiguation in some

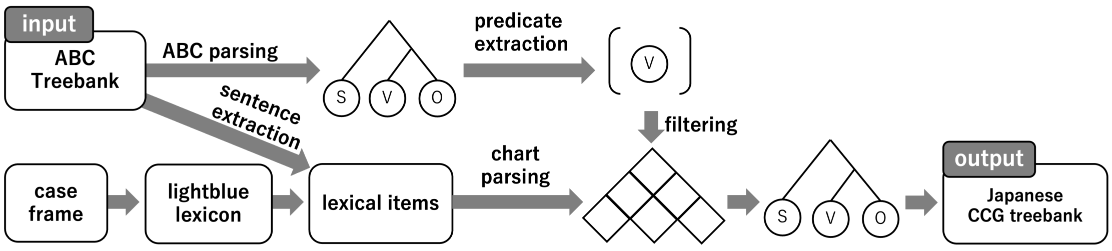


Figure 3: The reforging algorithm

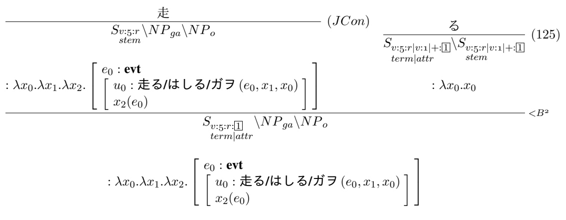


Figure 4: A lightblue output tree

contexts.

3 The Reforging Process

The aim of this research is to construct a linguistically valid Japanese CCG treebank with detailed syntactic features. Therefore, we propose a method for constructing a Japanese CCG treebank by combining the positive aspects of ABCTreebank, in which argument structures are manually annotated, with *lightblue*'s ability to provide CCG trees with detailed syntactic features. The proposed method decomposes ABCTreebank and reconstructs it using *lightblue*, a method we call "reforging". Note that "reforging" is not a commonly used linguistics term, but decomposing and reconstructing a treebank using a parser is a novel approach, so we assigned this name for convenience.

One approach to constructing a new CCG treebank would be to automatically correct Japanese CCGbank output. However, one difference between CCG and CFG is that modifying part of a CCG tree would require recalculating all subsequent calculations. Since those recalculations would be almost as costly as reparsing the entire sentence, this study aims to construct a treebank using reforging.

The reforging process has three steps:

1. Extract of predicates from ABCTreebank
2. Filter the *lightblue* lexicon chart
3. Reconstruct the treebank

Figure 3 shows each process in detail. A parser decomposes a given ABCTreebank input and its *lightblue* reconstruction is output as a treebank.

3.1 Predicate Extraction from ABCTreebank

Predicate extraction starts with ABC parsing, which gives tree-structured data. We then extract the predicate information from ABCTreebank as a list of tuples with the following four elements:

1. The phonetic form of the predicate
2. The syntactic category
3. The starting position in the sentence
4. The ending position in the sentence

(3) 人が 集まる
Hito-ga atumaru
People-NOM gather.
'People gather.' [5]

For example, in the case of sentence (3), the ABCTreebank syntactic structure is represented as shown in Figure 5. The predicate contained in this sentence is only "集まる(gather)", so a list of length 1 is extracted as [("集まる" $PP_s\backslash S_m, 2, 4$)].

3.2 Lexicon Chart Filtration

Filtration starts by extracting from the *lightblue* lexicon the lexical items of all substring combinations that exist in the ABCTreebank sentence. After obtaining those lexical items is chart parsing using the lexical items extracted in the previous step. Left-corner chart parsing is performed in *lightblue*, calculating node data while building a syntactic structure of the word by combining daughter nodes. Table 1 shows the node data structure. The next step is chart filtration, where, we filter the chart with the argument-structure information of the verb extracted from ABCTreebank.

[5]ABCTreebank ID: 3_textbook_kisonihongo

Data	Example
Rule symbol	<B2 (backward function composition rule)
Phonetic form	走る
Syntactic category	$S_{[v:5:r][stem]}\backslash NP_{ga}\backslash NP_o$
Semantic representation	$\lambda x0.\lambda x1.\lambda x2.(e0 : evt) \times (u0.\text{走る}\ (e0, x1, x0)) \times 2(e0)$
Signature	[走る:$(x0 : entity) \rightarrow (x1 : entity) \rightarrow (e0 : evt) \rightarrow type$]
Daughter nodes	node of 走(`run`) and node of る(`PRS`)
Score	[1.00]
Lexical entry source	JCon

Table 1: Data structure for 走る (run) node data

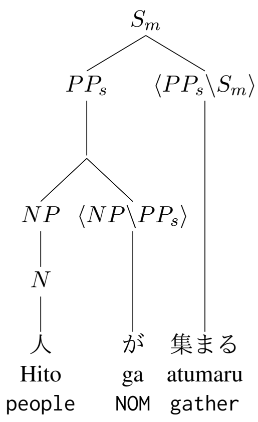

S_m
PP_s
$\langle PP_s\backslash S_m\rangle$
NP
$\langle NP\backslash PP_s\rangle$
N
人
Hito
people
が
ga
NOM
集まる
atumaru
gather

Figure 5: ABCTreebank syntactic structure of sentence (3) in ABCTreebank

Genre	Sentences	Len-50+ sentences	Reforged trees
aozora	1773	590	1183
bible	1652	220	1430
book_expert	50	4	41
dict_lexicon	2640	4	2636
diet_kaigiroku	486	112	374
fiction	921	44	877
law	337	128	209
misc	335	59	276
news	443	103	340
non-fiction	223	87	126
spoken	570	11	559
ted_talk	605	54	551
text-book	4880	10	4870
wikipedia	222	51	171
Total	15137	1482	13653

Table 2: Reforged treebank data

The filtering algorithm first extracts the phonetic form and predicate's syntactic category from ABCTreebank. The syntactic category is based on the ABCTreebank definition and needs to be converted to the *lightblue* definition for comparison with the *lightblue* category in the next step, which is to obtain lexical items of *lightblue* predicates. To obtain lexical items in *lightblue*, we use the predicate tuple data extracted from ABCTreebank. We extract the syntactic features of those lexical items that having the same phonetic form as the predicate extracted from ABCTreebank. In this study, the argument structures of adjectives and nominal predicates did not need to be filtered, so we excluded adjectives and nominal predicates. Valid and detailed syntactic information can be constructed by combining the the argument structure converted from ABCTreebank and the syntactic features extracted from *lightblue*.

3.3 Treebank Reconstruction

We randomly sampled five files from each of the 14 ABCTreebank genres and parsed them using *lightblue* and filtered charts. However, there are fewer than five ABCTreebank files for each of the bible, dict, fiction, and law genres, so we used all data in those genres.

The computational complexity of *lightblue* chart-parsing is $O(n^3)$, so CCG parsing of long sentences takes a long time. We thus limited sentences to fifty or fewer characters. Even with this restriction, 90.19% of sentences could be converted to CCG trees, which is sufficient coverage. Table 2 shows the statistics for the reforged treebank data. From left to right, the table shows the number of ABCTreebank sentences for each genre, the number of sentences with more than 50 characters, the number of CCG trees obtained by reforging, and the percentage of trees that could be converted to CCG trees by reforging from ABCTreebank data. We obtained 13,653 trees in total.

4 Discussion

Figure 6 shows the syntactic structure tree of the sentence (3) before filtering the *lightblue* chart, and Figure 7 shows the tree after filtering.

The *lightblue* lexical items for the predicate *gather* are the entries having ga-case and ni-case NPs as arguments. Therefore, *gather* was analyzed as having not only ga-case NP but also ni-case NP as arguments using the chart before filtering. However, the verb *gather* does not necessarily have the ni-case NP as an argument. Especially in sentence (3), it is reasonable to assume that *gather* has only ga-case NP as the argument. By overwriting the partial syntactic structure of the predicate *gather* having ga-case and ni-case NPs as arguments with a lexical entry having only ga-case NP as the argument, it became possible to convert ABCTreebank sentences to linguistically valid CCG syntactic structures. *lightblue* output also contains detailed syntactic features and Dependent Type Semantics (DTS; Bekki and Mineshima, 2017) representations.

4.1 Passive and Causative Sentences

As introduced in Section 2.2, Japanese CCGbank is based on the incorrect analysis for passive and causative sentences. To show how this study improved this issue, we discuss the passive sentences in the constructed treebank. The treebank constructed in this study includes the following passive and causative sentences:

(4) 太郎は 先生に 絵を ほめら れ た
Taro-wa sensei-ni e-o homera re ta
Taro-NOM teacher-DAT picture-OBJ praise passive PST

'Taro was praised for his picture by his teacher.' [6]

(5) 私は 猫に 魚を 食べさ せ た
Watasi-wa neko-ni sakana-o tabesa se ta
I-NOM cat-DAT fish-OBJ eat causative PST

'I fed the cat fish.' [7]

The CCG tree output by *lightblue* is in the Appendix. Figure 10 shows the syntactic structure and semantic representation of sentence (4). The category $S\backslash NP_{ga}\backslash NP_{ni}\backslash(S\backslash NP_{ga})$ is assigned to the passive suffix *re*, and the argument structure of the predicate *homera* is changed from $S\backslash NP_{ga}\backslash NP_{o}$ to $S\backslash NP_{ga}\backslash NP_{ni}$, which is linguistically valid. Figure 11 shows the syntactic structure and semantic representation of the sentence (5). The causative suffix *se* is also linguistically valid, in the same manner as the passive suffix *re*. These outputs for passive and causative sentences show the improvement of the incorrect analysis in Japanese CCGBank.

[6] ABCTreebankID:413_textbook_kisonihongo
[7] ABCTreebankID: 693_textbook_purple_intermediate

5 Error Analysis

The reforging process is successful in some sentences, but errors can still occur due to factors such as incorrect argument structures in ABCTReebank or incorrect analysis of the adnominal clause in *lightblue*.

5.1 Incorrect ABCTreebank Argument Structures

ABCTreebank occasionally contains incorrect argument structure annotations, and using such incorrect argument structures for reforging can remove correct lexical items in *lightblue*. For example, ABCTreebank contains erroneous argument structures for sentence (6).

(6) 鈴木さんが 街で 旧友に 会っ た
Suzukisan-ga machi-de kyuyu-ni at ta
Mr.Suzuki-NOM town-LOC old friend-DAT meet PST

'Mr.Suzuki met an old friend in town' [8]

Figure 8 shows the syntactic structure of sentence (6) in ABCTreebank. ABCTreebank analyzed *old friend* as an adverb phrase and assigned the category $\langle\langle PP_s\backslash S_m\rangle/\langle PP_s\backslash S_m\rangle\rangle$, despite the category PP_{o2} being correct for an ni-case noun phrase.

5.2 Incorrect Analysis of the Adnominal Clause

Sentence (7) below is an example of a sentence containing a relative clause:

(7) 食べる ものも なけれ ば、 住む
Taberu mono-mo nakere ba , sumu
所も ない
tokoro-mo nai
Eat thing-NOM no CONJ, live place-NOM no.

'No food to eat, no place to live' [9]

[8] ABCTreebankID:132_textbook_kisonihongo
[9] ABCTreebankID:12_textbook_particles

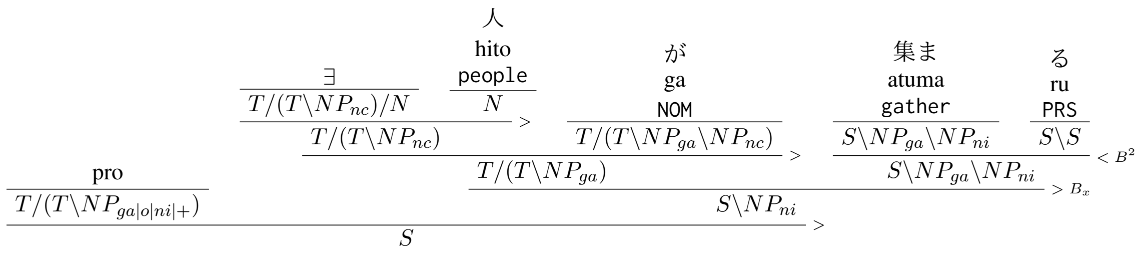


Figure 6: *lightblue* tree before reforging

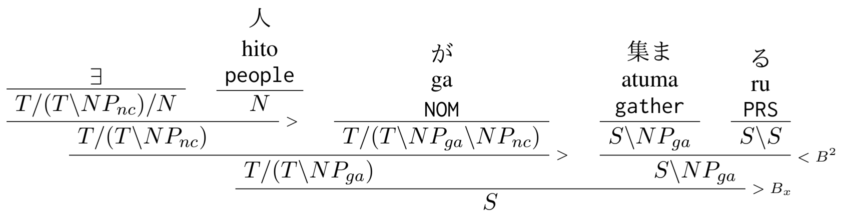


Figure 7: *lightblue* tree after reforging

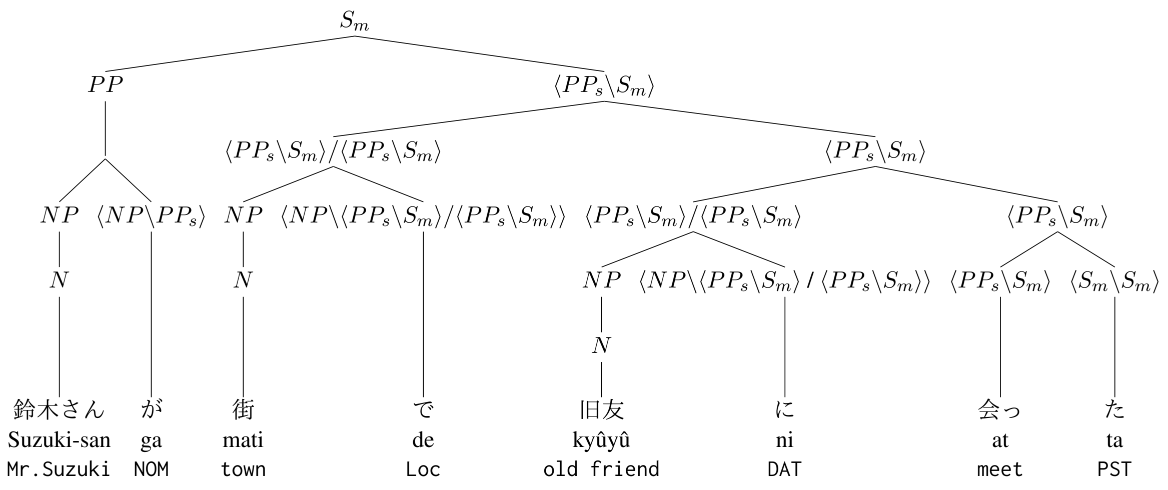
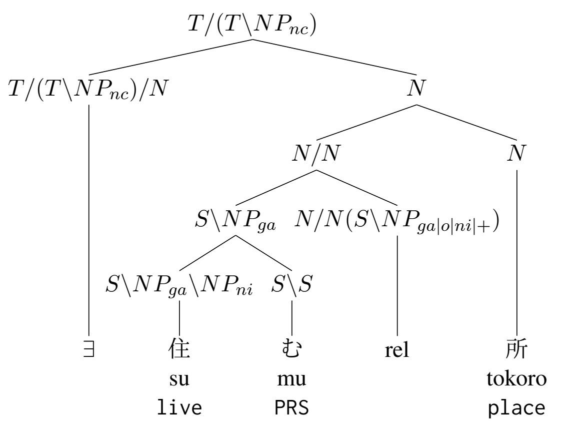


Figure 8: ABCTreebank syntactic structure of sentence (6)



$T/(T\backslash NP_{nc})$
$T/(T\backslash NP_{nc})/N$
N
N/N
N
$S\backslash NP_{ga}$
$N/N(S\backslash NP_{ga|o|ni|+})$
$S\backslash NP_{ga}\backslash NP_{ni}$
$S\backslash S$
∃
住
む
rel
所
su
mu
tokoro
live
PRS
place

Figure 9: Invalid output for an adnominal clause sentence

Figure 9 shows part of the tree output after reforging. In this tree, the predicate *live* takes the ga-case NP and becomes a relative clause, but cannot be interpreted as *the place is lived*. Thus, an external relation was analyzed as an internal relation.

6 Evaluation

Since the linguistic validity of constructed treebanks cannot be automatically evaluated, it is necessary to manually check one-by-one whether each syntactic structure and semantic representation is correctly obtained. However, the cost of manually evaluating the syntactic structures of CCG and DTS representations would be very high, and it was unfeasible to manually evaluate all of the 13,653 sentences constructed in this study. We thus manually evaluated 56 sentences, randomly sampling four sentences from each genre. We also evaluated the constructed treebanks by their conversion rates. As evaluation metrics for machine-learning-based CCG parsers such as depccg (Yoshikawa et al., 2017), lexical coverage, sentential coverage, and syntactic rule coverage are used. However, we did not use supervised learning methods for CCG parsing; instead, we performed rule-based conversions. Consequently, evaluations using unseen data were infeasible, so we used only conversion rates as an evaluation metric.

6.1 Conversion Rate

The conversion rate is the percentage of sentences fully converted to CCG trees. As a result of reforging, out of 13,655 sentences, 13,653 sentences were successfully converted, for a conversion rate of 99.9%.

	Metrics	Sentences
Syntactic Error	Syntactic category	18
	Compound verb	4
	Other syntactic error	30
Semantic Error		7
No Error		19

Table 3: Results from manual evaluations

6.2 Manual Evaluation

We manually evaluated 56 randomly sampled sentences, four from each genre. Manual evaluations considered whether the sentences had correct syntactic structures from three perspectives:

1. Whether the sentence was assigned a invalid syntactic category
2. Whether compound verbs are analyzed separately
3. Whether other syntactic errors are included

We also evaluated the validity of output DTS representations to see whether correct semantic representations are obtained from syntactic structures. Table 3 shows the evaluation results. Syntactically and semantically valid trees were produced for 19 of 56 sentences (33%). One of the most common observed errors was the invalid syntactic categories. In particular, we observed several cases where the word *ni*, which should be analyzed as a case-marking particle, was incorrectly analyzed as the stem of the verb *niru* which means "to boil." This occurred when reforging overwrote the syntactic category a predicate having *ni* case NP as an argument. In the future, it will be necessary to eliminate this error by making an exception for the word *ni*.
The incorrect reforging output can be categorized into the following four error cases:

1. The pre-reforging argument structure is incorrect, but the post-reforging argument structure is correct.
2. The pre-reforging argument structure is correct, but reforging results in an error
3. Both the pre-reforging and post-reforging argument structures are correct.
4. Both the pre-reforging and post-reforging argument structures are incorrect.

Reforging was successful in Case 1. Case 2 occurs when there are inaccuracies in ABCTreebank's argument structure. Case 3 signifies instances where reforging had no impact, while Case 4 involves errors originating from *lightblue* that cannot be resolved through reforging. A future goal is to address and improve errors related to the argument structures that occur in Case 2.

7 Conclusion

We proposed a reforging method for constructing linguistically valid Japanese CCG treebanks with detailed syntactic features. Our method obtained correct Japanese CCG syntactic structures to some extent. Our method assumes that ABCTreebank argument structures are valid because their syntactic structures are manually annotated. However, there is an upper bound on the validity of ABCTreebank argument structures. To obtain linguistically valid argument structures, our future work will consider combining ABCTreebank with other reliable resources. We also plan to improve our filtering algorithm, and improve *lightblue*'s parsing algorithm to better handle long sentences.

Acknowledgements

We thank the two anonymous reviewers for their helpful comments and feedback. This work was partially supported by JSPS KAKENHI grant number JP20K19868, Japan.

References

Daisuke Bekki. 2010. *Nihongo-Bunpoo-no KeisikiRiron - Katuyootaikei, Toogohantyuu, Imigoosei - (trans. ' Formal Japanese Grammar: the conjugation system, categorial syntax, and compositional semantics').* Kuroshio Publisher, Tokyo.

Daisuke Bekki and Ai Kawazoe. 2016. Implementing variable vectors in a CCG parser. In *Logical Aspects of Computational Linguistics. Celebrating 20 Years of LACL (1996–2016)*, pages 52–67, Berlin, Heidelberg. Springer Berlin Heidelberg.

Daisuke Bekki and Koji Mineshima. 2017. Context-passing and underspecification in dependent type semantics.

Daisuke Bekki and Hitomi Yanaka. 2023. Is Japanese CCGBank empirically correct? a case study of passive and causative constructions. In *Proceedings of the 21st International Workshop on Treebanks and Linguistic Theories (TLT, GURT/SyntaxFest 2023)*, pages 32–36, Washington, D.C. Association for Computational Linguistics.

Johan Bos, Cristina Bosco, and Alessandro Mazzei. 2010. Converting a dependency treebank to a categorial grammar treebank for italian.

Stephen A. Boxwell and Chris Brew. 2010. A pilot Arabic CCGbank. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Ted Briscoe and John Carroll. 2006. Evaluating the accuracy of an unlexicalized statistical parser on the PARC DepBank. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 41–48, Sydney, Australia. Association for Computational Linguistics.

Martin Forst. 2003. Treebank conversion - establishing a testsuite for a broad-coverage LFG from the TIGER treebank. In *Proceedings of 4th International Workshop on Linguistically Interpreted Corpora (LINC-03) at EACL 2003.*

Hiroki Hanaoka, Hideki Mima, and Jun'ichi Tsujii. 2010. A Japanese particle corpus built by example-based annotation. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Julia Hockenmaier. 2006. Creating a CCGbank and a wide-coverage CCG lexicon for German. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 505–512, Sydney, Australia. Association for Computational Linguistics.

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.

Daisuke Kawahara and Sadao Kurohashi. 2006. Case frame compilation from the web using high-performance computing. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy. European Language Resources Association (ELRA).

Yusuke Kubota, Koji Mineshima, Noritsugu Hayashi, and Shinya Okano. 2020. Development of a general-purpose categorial grammar treebank. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 5195–5201, Marseille, France. European Language Resources Association.

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

Michael Moortgat. 1997. Categorial type logics. In J. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*. Elsevier.

Glyn Morrill. 1994. *Type Logical Grammar: Categorial Logic of Signs*.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press.

Mark J. Steedman. 1996. *Surface Structure and Interpretation*. The MIT Press, Cambridge.

Sumire Uematsu, Takuya Matsuzaki, Hiroki Hanaoka, Yusuke Miyao, and Hideki Mima. 2013. Integrating multiple dependency corpora for inducing wide-coverage Japanese CCG resources. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1042–1051, Sofia, Bulgaria. Association for Computational Linguistics.

David Vadas and James Curran. 2007. Adding noun phrase structure to the Penn Treebank. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 240–247, Prague, Czech Republic. Association for Computational Linguistics.

Masashi Yoshikawa, Hiroshi Noji, and Yuji Matsumoto. 2017. A* CCG parsing with a supertag and dependency factored model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 277–287, Vancouver, Canada. Association for Computational Linguistics.

A Appendix

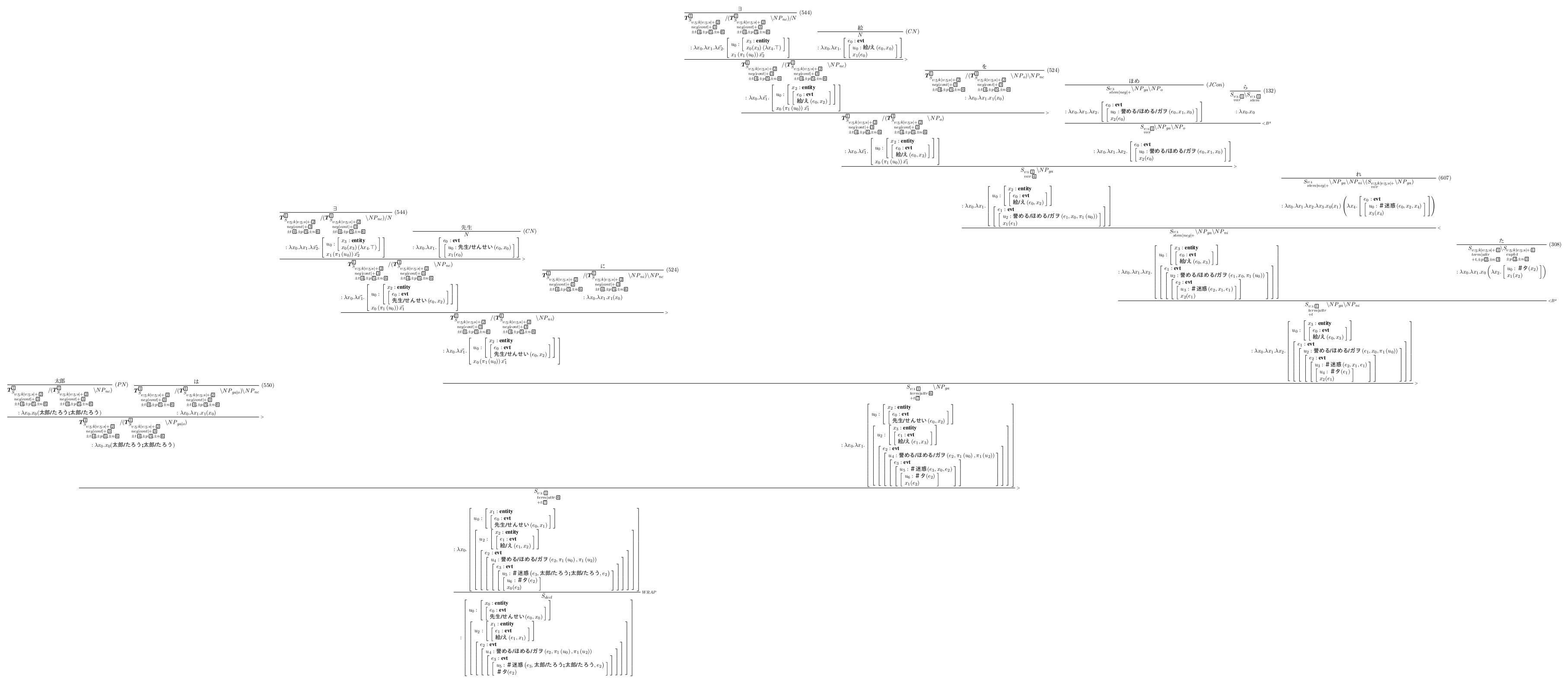


Figure 10: CCG tree of the passive sentence constructed by lightblue

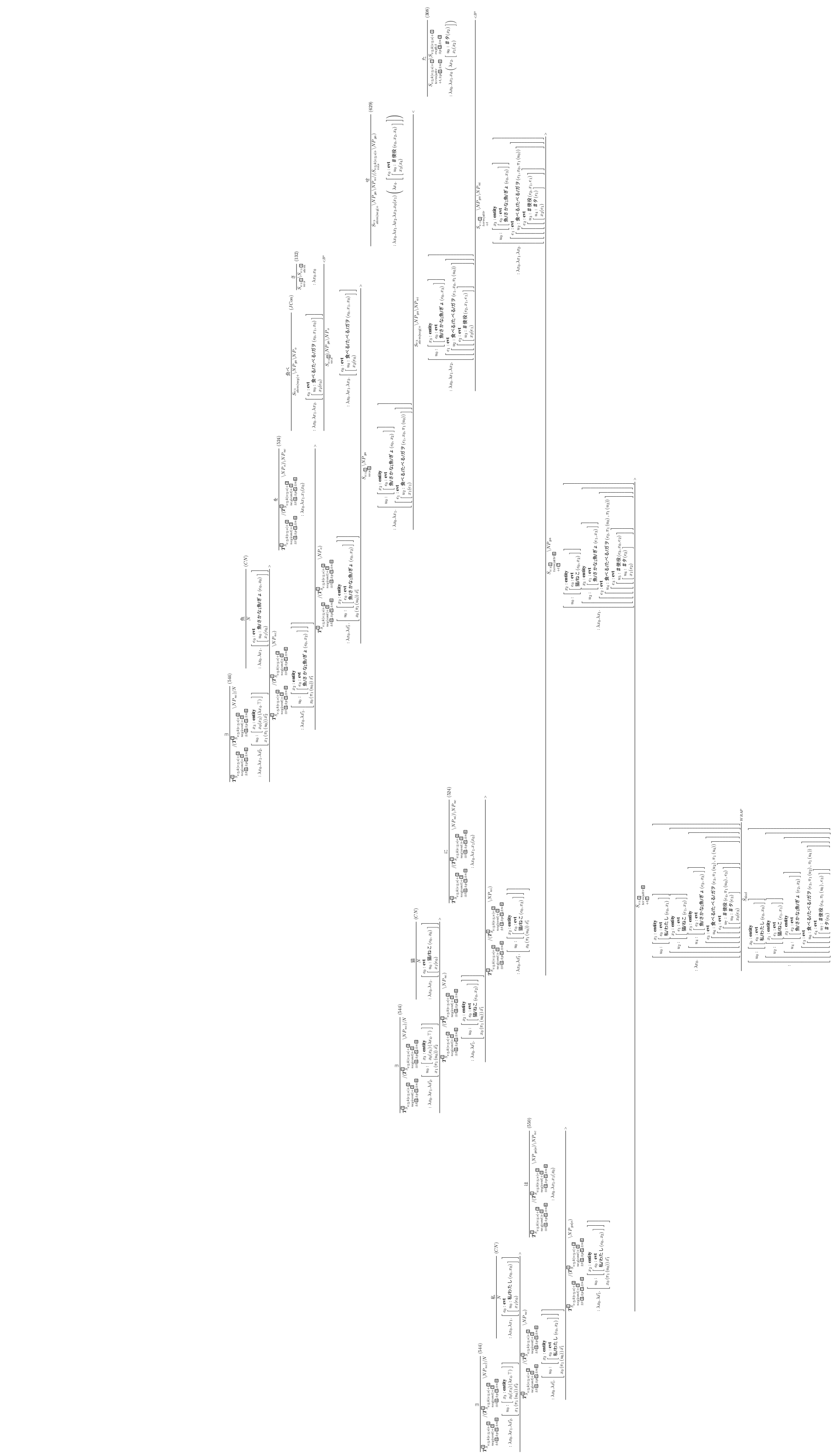


Figure 11: CCG tree of the causative sentence constructed by lightblue