A Collaborative Annotation between Human Annotators and a Statistical Parser

Shun'ya Iwasawa **Hiroki Hanaoka** **Takuya Matsuzaki**
University of Tokyo
Tokyo, Japan
{iwasawa,hkhana,matuzaki}@is.s.u-tokyo.ac.jp

Yusuke Miyao
National Institute of Informatics
Tokyo, Japan
yusuke@nii.ac.jp

Jun'ichi Tsujii
Microsoft Research Asia
Beijing, P.R.China
jtsujii@microsoft.com

Abstract

We describe a new interactive annotation scheme between a human annotator who carries out simplified annotations on CFG trees, and a statistical parser that converts the human annotations automatically into a richly annotated HPSG treebank. In order to check the proposed scheme's effectiveness, we performed automatic pseudo-annotations that emulate the system's idealized behavior and measured the performance of the parser trained on those annotations. In addition, we implemented a prototype system and conducted manual annotation experiments on a small test set.

1 Introduction

On the basis of the success of the research on the corpus-based development in NLP, the demand for a variety of corpora has increased, for use as both a training resource and an evaluation data-set. However, the development of a richly annotated corpus such as an HPSG treebank is not an easy task, since the traditional two-step annotation, in which a parser first generates the candidates and then an annotator checks each candidate, needs intensive efforts even for well-trained annotators (Marcus et al., 1994; Kurohashi and Nagao, 1998). Among many NLP problems, adapting a parser for out-domain texts, which is usually referred to as domain adaptation problem, is one of the most remarkable problems. The main cause of this problem is the lack of corpora in that domain. Because it is difficult to prepare a sufficient corpus for each domain without reducing the annotation cost, research on annotation methodologies has been intensively studied.

There has been a number of research projects to efficiently develop richly annotated corpora with the help of parsers, one of which is called a discriminant-based treebanking (Carter, 1997). In discriminant-based treebanking, the annotation process consists of two steps: a parser first generates the parse trees, which are annotation candidates, and then a human annotator selects the most plausible one. One of the most important characteristics of this methodology is to use easily-understandable questions called discriminants for picking up the final annotation results. Human annotators can perform annotations simply by answering those questions without closely examining the whole tree. Although this approach has been successful in breaking down the difficult annotations into a set of easy questions, specific knowledge about the grammar, especially in the case of a deep grammar, is still required for an annotator. This would be the bottleneck to reduce the cost of annotator training and can restrict the size of annotations.

Interactive predictive parsing (Sánchez-Sáez et al., 2009; Sánchez-Sáez et al., 2010) is another approach of annotations, which focuses on CFG trees. In this system, an annotator revises the currently proposed CFG tree until he or she gets the correct tree by using a simple graphical user interface. Although our target product is a more richly annotated treebanks, the interface of CFG can be useful to develop deep annotations such as HPSG features by cooperating with a statistical deep parser. Since CFG is easier to understand than HPSG, it can reduce the cost of annotator training; non-experts can perform annotations without decent training. As a result, crowd-sourcing or similar approach can be adopted and the annotation process would be accelerated.

Before conducting manual annotation, we simulated the annotation procedure for validating our system. In order to check whether the CFG-based annotations can lead to sufficiently accurate HPSG annotations, several HPSG treebanks were created with various qualities of CFG and evaluated by their HPSG qualities.

We further conducted manual annotation experiments by two human annotators to evaluate the efficiency of the annotation system and the accuracy of the resulting annotations. The causes of annotation errors were analyzed and future direction of the further development is discussed.

2 Statistical Deep Parser

2.1 HPSG

Head-Driven Phrase Structure Grammar (HPSG) is one of the lexicalized grammatical formalisms, which consists of lexical entries and a collection of schemata. The lexical entries represent the syntactic and semantic characteristics of words, and the schemata are the rules that construct larger phrases from smaller phrases. Figure 1 shows the mechanism of the bottom-up HPSG parsing for the sentence "Dogs run." First, a lexical entry is assigned to each word, and then, the lexical signs for "Dogs" and "run" are combined by Subject-Head schema. In this way, lexical signs and phrasal signs are combined until the whole sentence becomes one sign. Compared to Context Free Grammar (CFG), since each sign of HPSG has rich information about the phrase, such as subcategorization frame or predicate-argument structure, a corpus annotated in an HPSG manner is more difficult to build than CFG corpus. In our system, we aim at building HPSG treebanks with low-cost in which even non-experts can perform annotations.

2.2 HPSG Deep Parser

The Enju parser (Ninomiya et al., 2007) is a statistical deep parser based on the HPSG formalism. It produces an analysis of a sentence that includes the syntactic structure (i.e., parse tree) and the semantic structure represented as a set of predicate-argument dependencies. The grammar design is based on the standard HPSG analysis of English (Pollard and Sag, 1994). The parser finds a best parse tree scored by a maxent disambiguation model using a CKY-style algorithm and beam search. We used a toolkit distributed with the Enju parser for extracting a HPSG lexicon from a PTB-style treebank. The toolkit initially converts the PTB-style treebank into an HPSG treebank and then extracts the lexicon from it. The HPSG treebank converted from the test section is also used as the gold standard in the evaluation.

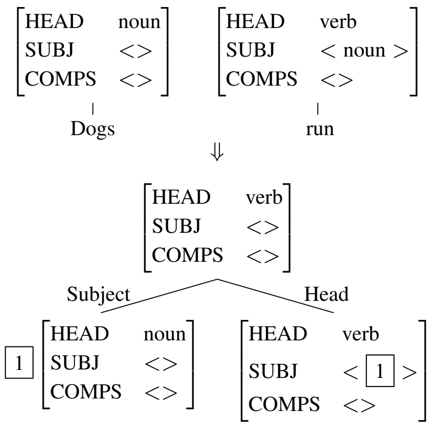

Figure 1: Example of HPSG parsing for "Dogs run."

2.3 Evaluation Metrics

In the experiments shown below, we evaluate the accuracy of an annotation result (i.e., an HPSG derivation on a sentence) by evaluating the accuracy of the semantic description produced by the derivation, as well as a more traditional metrics such as labeled bracketing accuracy of the tree structure. Specifically, we used labeled and unlabeled precision/recall/F-score of the predicate-argument dependencies and the labeled brackets compared against a gold-standard annotation obtained by using the Enju's treebank conversion tool. A predicate-argument dependency is represented as a tuple of $\langle w_p, w_a, r\rangle$, where w_p is the predicate word, w_a is the argument word, and r is the label of the predicate-argument relation, such as `verb-ARG1` (semantic subject of a verb) and `prep-MOD` (modifiee of a prepositional phrase). As for the bracketing accuracies, the label of a bracket is obtained by projecting the sign corresponding to the phrase into a simple phrasal labels such as S, NP, and VP.

3 Proposed Annotation System

In our system, a human annotator and a statistical deep parser cooperate to build a treebank. Our system uses CFG as user interface and bridges a gap between CFG and HPSG with a statistical CKY parser. Following the idea of the discriminant-based treebanking model, the parser first generates candidate trees and then an annotator selects the correct tree in the form of a packed forest. For selecting the correct tree, the annotator only edits a CFG tree projected from an HPSG tree through pre-defined set of operations, to eventually give the constraints onto HPSG trees. This is why annotators can annotate HPSG trees without HPSG knowledge. The current system is implemented based on the following client-server model.

3.1 Client: Annotator Interface

The client-side is an annotator's interface implemented with Ajax technique, on which annotator's revision is carried out through Web-Browser. When the client-side receives the data of the current best tree from the server-side, it shows an annotator the CFG representation of the tree. Then, an annotator adds revisions to the CFG tree using the same GUI, until the current best tree has the CFG structure that exactly matches the annotators' interpretation of the sentence. Finally, the client-side sends the annotator's revision as a CGI query to the server. Based on interactive predicative parsing system, two kinds of operations are implemented in our system: "span modification" and "label substitution", here abbreviated as "S" and "L" operations:

"S" operation *modify_span*$(left, right)$
An annotator can specify that a constituent in the tree after user's revision must match a specified span, by sequentially clicking the leaf nodes at the left and right boundaries.

"L" operation *modify_label*$(pos, label)$
An annotator can specify that a constituent in the tree after user's revision must match a specified label, by inputting a label and clicking the node position.

In addition to "S" and "L" operations, one more operation, "tree fixation", abbreviated "F", is implemented for making annotation more efficient. Our system computes the best tree under the current constraints, which are specified by the "S" and "L" operations that the annotator has given so far. It means other parts of the tree that are not constrained may change after a new operation by the annotator. This change may lead to a structure that the annotator does not want. To avoid such unexpected changes, an annotator can specify a subtree which he or she does not want to change by "tree fixation" operation:

"F" operation *fix_tree*$(pos = i)$
An annotator can specify a subtree as correct and not to be changed. The specified subtree does not change and always appears in the best tree.

3.2 Server: Parsing Constraints

In our annotation system, the server-side carries out the conversion of annotator's constraints into HPSG grammatical constraints on CKY chart and the re-computation of the current best tree under the constraints added so far. The server-side works in the following two steps. The first step is the conversion of the annotator's revision into a collection of dead edges or dead cells; a dead edge means the edge must not be a part of the correct tree, and a dead cell means all edges in the cell are dead. As mentioned in the background section, Enju creates a CKY chart during the parsing where all the terminal and non-terminal nodes are stored with the information of its sign and links to daughter edges. In our annotation system, to change the best tree according to the annotator's revision, we determine whether each edge in the chart is either alive or dead. The server-side re-constructs the best tree under the constraints that all the edges used in the tree are alive. The second step is the computation of the best tree by reconstructing the tree from the chart, under the constraint that the best tree contains only the alive edges as its subconstituents. Re-construction includes the following recursive process:

1. Start from the root edge.

2. Choose the link which has the highest probability among the links and whose daughter edges are all alive.
3. If there is such a link, recursively carry out the process for the daughter edge.
4. If all the links from the edge are dead, go back to the previous edge.

Note that our system parses a sentence only once, the first time, instead of re-parsing the sentence after each revision. Now, we are going to list the revision operations again and explain how the operations are interpreted as the constraints in the CKY chart. In the description below, `label`(x) means the CFG-symbol that corresponds to edge x. Note that there is in principle an infinite variety of possible HPSG signs. The label function maps this multitude of signs onto a small set of simple CFG nonterminal symbols.

"S" operation *span*$(left = i, right = j)$
When the revision type is "S" and the left and right boundary of the specified span is i and j in the CGI query, we add the cells which satisfy the following formula to the list of dead edges. Suppose the sentence length is L, then the set of new dead cells is defined as:

$$\{\texttt{cell}(a,b) \mid \begin{array}{c} 0 \le a < i, \\ i \le b < j \end{array} \}$$
$$\bigcup \{\texttt{cell}(c,d) \mid \begin{array}{c} i+1 \le c \le j, \\ j+1 \le d \le n \end{array} \},$$

where the first set means the inhibition of the edges that span across the left boundary of the specified span. The second set means a similar conditions for the right span.

"L" operation *fix_label*$(position = i, label = l)$
When the revision type is "L", the node position is i and the label is l in the CGI query, we determine the set of new dead edges and dead cells as follows:
1. let `cell`(a,b) = the cell including i
2. mark those cells that are generated by `span`(a,b) as defined above to be dead, and
3. for each edge e' in `cell`(a,b), mark e' to be dead if `label`$(e') \neq l$

"F" operation *fix_tree*$(position = i)$

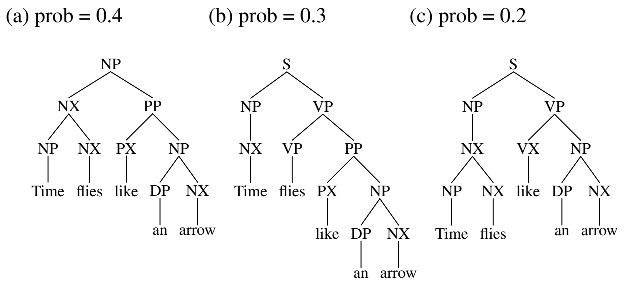

Figure 2: Three parse tree candidates of "Time flies like an arrow."

When the revision type is "F" and the target node position is i in the CGI query, we carry out the following process to determine the new dead edges and cells:
1. for each edge e in the subtree rooted at node i,
2. let `cell`(a,b) = the cell including e
3. mark those cells that are generated by `span`(a,b) as defined above to be dead
4. for each edge e' in `cell`(a,b), mark e' to be dead if `label`$(e') \neq$ `label`(e)

The above procedure adds the constraints so that the correct tree includes a subtree that has the same CFG-tree representation as the subtree rooted at i in the current tree.

Finally we show how the best tree for the sentence "Time flies like an arrow." changes with the annotator's operations. Let us assume that the chart includes the three trees shown (in the CFG representation) in (Figure 2), and that there are no dead edges. Let us further assume that the probability of each tree is as shown in the figure and hence the current best tree is (a). If the annotator wants to select (b) as the best tree, s/he can apply "L" operation on the root node. The operation makes some of the edges dead, which include the root edge of tree (a) (see Figure 3). Accordingly, the best tree is now selected from (b), (c), etc., and tree (b) will be selected as the next best tree.

4 Validation of CFG-based Annotation

Because our system does not present HPSG annotations to the annotators, there is a risk that HPSG annotations are wrong even when their projections to CFG trees are completely correct. Our expecta-

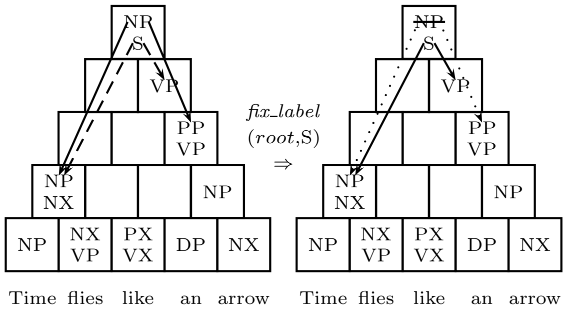

Figure 3: Chart constraints by "L" operation. Solid lines represent the link of the current best tree and dashed lines represent the second best one. Dotted lines stand for an unavailable link due to the death of the source edge.

tion is that the stochastic model of the HPSG parser properly resolves the remaining ambiguities in the HPSG annotation within the constraints given by a part of the CFG trees. In order to check the validity of this expectation and to measure to what extent the CFG-based annotations can achieve correct HPSG annotations, we performed a pseudo-annotation experiment.

In this experiment, we used bracketed sentences in the Brown Corpus (Kučera and Francis, 1967), and a court transcript portion of the Manually Annotated Sub-Corpus (MASC) (Ide et al., 2010). We automatically created HPSG annotations that mimic the annotation results by an ideal annotator in the following four steps. First, HPSG treebanks for these sentences are created by the treebank conversion program distributed with the Enju parser. This program converts a syntactic tree annotated by Penn Treebank style into an HPSG tree. Since this program cannot convert the sentences that are not covered by the basic design of the grammar, we used only those that are successfully converted by the program throughout the experiments and considered this converted treebank as the gold-standard treebank for evaluation. Second, the same sentences are parsed by the Enju parser and the results are compared with the gold-standard treebank. Then, CFG-level differences between the Enju parser's outputs and the gold-standard trees are translated into operation sequences of the annotation system. For example, "L" operation of NX → VP at the root node is obtained in the case of Figure 4. Finally, those operation sequences are executed on the annotation system and HPSG annotations are produced.

	total size	ave. s. l.	convertible
Brown	24,243	18.94	22,214
MASC	1,656	14.81	1,353

Table 1: Corpus and experimental data information (s. l. means "sentence length.")

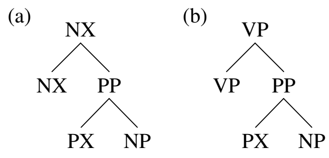

Figure 4: CFG representation of parser output (a) and gold-standard tree (b)

4.1 Relationship between CFG and HPSG Correctness

We evaluated the automatically produced annotations in terms of three measures: the labeled bracketing accuracies of their projections to CFG trees, the accuracy of the HPSG lexical entry assignments to the words, and the accuracy of the semantic dependencies extracted from the annotations. The CFG-labeled bracketing accuracies are defined in the same way as the traditional PARSEVAL measures. The HPSG lexical assignment accuracy is the ratio of words to which the correct HPSG lexical entry is assigned, and the semantic dependency accuracy is defined as explained in Section 2.3. In this experiment, we cut off sentences longer than 40 words for time reasons. We split the Brown Corpus into three parts: training, development test and evaluation, and evaluated the automatic annotation results only for the training portion.

We created three sets of automatic annotations as follows:

Baseline No operation; default parsing results are considered as the annotation results.

S-full Only "S" operations are used; the tree structures of the resulting annotations should thus be identical to the gold-standard annotations.

SL-full "S" and "L" operations are used; the labeled tree structures of the resulting annotations should thus be identical to the gold-standard annotations.

Before showing the evaluation results, splitting of the data should be described here. Our system assumes that the correct tree is included in the parser's

CKY chart; however, because of the beam-search limitation and the incomplete grammar coverage, it does not always hold true. In this paper, such situations are called "out-chart". Conversely, the situations in which the parser does include the correct tree in the CKY chart are "in-chart". The results of "in-chart" are here considered to be the results in the ideal situation of the perfect parser. In our experimental setting, the training portion of the Brown Corpus has 10,576 "in-chart" and 7,208 "out-chart" sentences, while the MASC portion has 864 "in-chart" and 489 "out-chart" sentences (Table 2). Under "out-chart" situations, we applied the operations greedily for calculating S-full and SL-full; that is, all operations are sequentially applied and an operation is skipped when there are no HPSG trees in the CKY chart after applying that operation.

Table 3 shows the results of our three measures: the CFG tree bracketing accuracy, the accuracy of HPSG lexical entry assignment and that of the semantic dependency. In both of S-full and SL-full, the improvement from the baseline is significant. Especially, SL-full for "in-chart" data has almost complete agreement with the gold-standard HPSG annotations. The detailed figures are shown in Table 4. Therefore, we can therefore conclude that high quality CFG annotations lead to high quality HPSG annotations when the are combined with a good statistical HPSG parser.

4.2 Domain Adaptation

We evaluated the parser accuracy adapted with the automatically created treebank on the Brown Corpus. In this experiment, we used the adaptation algorithm by (Hara et al., 2007), with the same hyper-parameters used there. Table 5 shows the result of the adapted parser. Each line of this table stands for the parser adapted with different data. "Gold" is the result adapted on the gold-standard annotations, and "Gold (only covered)" is that adapted on the gold data which is covered by the original Enju HPSG grammar that was extracted from the WSJ portion of the Penn Treebank. "SL-full" is the result adapted on our automatically created data. "Baseline" is the result by the original Enju parser, which is trained only on the WSJ-PTB and whose grammar was extracted from the WSJ-PTB. The table shows SL-full slightly improves the baseline results, which indicates our annotation system can be useful for domain adaptation. Because we used mixed data of "in-chart" and "out-chart" in this experiment, there still is much room for improvement by increasing the ratio of the "in-chart" sentences using a larger beam-width.

		#operations				Time
		S	L	F	Avg.	
Brown	A. 1	122	1	0	1.19	43.32
	A. 2	91	4	1	0.94	41.77
MASC	A. 1	275	2	5	2.76	33.33
	A. 2	52	2	0	0.51	35.13

Table 6: The number of operations and annotation time by human annotators. "Annotator" is abbreviated as A. Avg. is the average number of operations per sentence and Time is annotation time per sentence [sec.].

5 Interactive Annotation on a Prototype-system

We developed an initial version of the annotation system described in Section 3, and annotated 200 sentences in total on the system. Half of the sentences were taken from the Brown corpus and the other half were taken from a court-debate section of the MASC corpus. All of the sentences were annotated twice by two annotators. Both of the annotators has background in computer science and linguistics.

Table 6 shows the statistics of the annotation procedures. This table indicates that human annotators strongly prefer "S" operation to others, and that the manual annotation on the prototype system is at least comparable to the recent discriminant-based annotation system by (Zhang and Kordoni, 2010), although the comparison is not strict because of the difference of the text.

Table 7 shows the automatic evaluation results. We can see that the interactive annotation gave slight improvements in all accuracy metrics. The improvements were however not as much as we desired.

By classifying the remaining errors in the annotation results, we identified several classes of major errors:

1. Truly ambiguous structures, which require the context or world-knowledge to correctly resolve them.

	in	out	in+out
Brown (train.)	10,576 / 10,394	7,190 / 6,464	17,766 / 16,858
MASC	864 / 857	489 / 449	1,353 / 1,306

Table 2: The number of "in-chart" and "out-chart" sentences (total / 1-40 length)

		in	out	in+out
Brown	SL-full	100.00 / 99.31 / 99.60	88.67 / 83.95 / 82.00	94.91 / 92.21 / 92.24
	S-full	98.46 / 96.64 / 96.83	89.60 / 82.02 / 81.20	94.48 / 89.88 / 90.29
	Baseline	92.39 / 92.69 / 90.54	82.10 / 78.38 / 73.80	87.78 / 86.07 / 83.54
MASC	SL-full	100.00 / 99.13 / 99.30	85.91 / 80.75 / 78.80	93.38 / 90.55 / 91.02
	S-full	98.71 / 96.88 / 96.73	86.95 / 79.14 / 77.43	93.18 / 88.60 / 88.93
	Baseline	93.98 / 93.51 / 91.56	80.00 / 75.89 / 72.22	87.43 / 85.30 / 83.75

Table 3: Evaluation of the automatic annotation sets. Each cell has the score of CFG F1 / Lex. Acc. / Dep. F1.

	CFG tree accuracy	
	Brown	MASC
A. 1	90.55 / 90.83 / 90.69	90.62 / 90.80 / 90.71
A. 2	91.01 / 91.09 / 91.05	91.01 / 91.09 / 91.05
Enju	89.70 / 89.74 / 89.72	90.02 / 90.20 / 90.11
	PAS dependency accuracy	
	Brown	MASC
A. 1	87.48 / 87.55 / 87.52	86.02 / 86.02 / 86.02
A. 2	88.42 / 88.27 / 88.34	85.28 / 91.01 / 85.32
Enju	87.12 / 86.91 / 87.01	84.81 / 84.26 / 84.53

Table 7: Automatic evaluation of the annotation results (LP / LR / F1)

	CFG tree accuracy	
	in-chart	out-chart
A. 1	94.52 / 94.65 / 94.58	83.95 / 84.44 / 84.19
A. 2	95.07 / 95.14 / 95.10	84.22 / 84.32 / 84.27
Enju	94.44 / 94.37 / 94.40	81.81 / 82.00 / 81.90
	PAS dependency accuracy	
	in-chart	out-chart
A. 1	92.85 / 92.85 / 92.85	77.47 / 77.65 / 77.56
A. 2	93.34 / 93.34 / 93.34	79.17 / 78.80 / 78.98
Enju	92.73 / 92.73 / 92.73	76.57 / 76.04 / 76.30

Table 8: Automatic evaluation of the annotation results (LP/LR/F1); in-chart sentences (left-column) and out-chart sentences (right column) both from Brown

2. Purely grammar-dependent analyses, which require in-depth knowledge of the specific HPSG grammar behind the simplified CFG-tree representation given to the annotators.
3. Discrepancy between human intuition and the convention in the HPSG grammar introduced by the automatic conversion.
4. Apparently wrong analysis left untouched due to the limitation of the annotation system.

We suspect some of the errors of type 1 have been caused by the experimental setting of the annotation; we gave the test sentences randomly drawn from the corpus in a randomized order. This would have made it difficult for the annotators to interpret the sentences correctly. We thus expect this kind of errors would be reduced by doing the annotation on a larger chunk of text.

The second type of the errors are due to the fact that the annotators are not familiar with the details of the Enju English HPSG grammar. For example, one of the annotators systematically chose a structure like (NP (NP a cat) (PP on the mat)). This structure is however always analysed as (NP a (NP' cat (PP on the mat))) by the Enju grammar. The style of the analysis implemented in the grammar thus sometimes conflicts with the annotators' intuition and it introduces errors in the annotation results.

Our intention behind the design of the annotation system was to make the annotation system more accessible to non-experts and reduce the cost of the annotation. To reduce the type 2 errors, rather than the training of the annotators for a specific grammar, we plan to introduce another representation system in which the grammar-specific conventions become invisible to the annotators. For example, the above-shown difference in the bracketing structures of a determiner-noun-PP sequence can be hidden by showing the noun phrase as a ternary branch on the three children: (NP a cat (PP on the mat)).

The third type of the errors are mainly due to the rather arbitrary choice of the HPSG analysis introduced through the semi-automatic treebank conversion used to extract the HPSG grammar. For instance, the Penn Treebank annotates a structure including an adverb that intervenes an auxiliary verb

	Lex-Acc	Dep-LP	Dep-LR	Dep-UP	Dep-UR	Dep-F1	Dep-EM
Brown	99.26	99.61	99.59	99.69	99.67	99.60	95.80
MASC	99.13	99.26	99.33	99.42	99.49	99.30	95.68

Table 4: HPSG agreement of SL-full for “in-chart” data (EM means “Exact Match.”)

	LP	LR	UP	UR	F1	EM
Gold	85.62	85.41	89.70	69.47	85.51	45.07
Gold (only covered)	84.32	84.01	88.72	88.40	84.17	42.52
SL-full	83.27	82.88	87.93	87.52	83.08	40.19
Baseline	82.64	82.20	87.50	87.03	82.42	37.63

Table 5: Domain Adaptation Results

and a following verb as in (VP is (ADVP already) installed). The attachment direction of the adverb is thus left unspecified. Such structures are however indistinguishably transformed to a binary structure like (VP (VP’ is already) installed) in the course of the conversion to HPSG analysis since there is no way to choose the proper direction only with the information given in the source corpus. This design could be considered as a best-effort, systematic choice under the insufficient information, but it conflicts with the annotators’ intuition in some cases.

We found in the annotation results that the annotators have left apparently wrong analyses on some sentences, either those remaining from the initial output proposed by the parser or a wrong structure appeared after some operations by the annotators (error type 4). Such errors are mainly due to the fact that for some sentences a correct analysis cannot be found in the parser’s CKY chart. This can happen either when the correct analysis is not covered by the HPSG grammar, or the correct analysis has been pruned by the beam-search mechanism in the parser. To correct a wrong analysis from the insufficient grammar coverage, an expansion of the grammar is necessary, either in the form of the expansion of the lexicon, or an introduction of a new lexical type. For the other errors from the beam-search limitation, there is a chance to get a correct analysis from the parser by enlarging the beam size as necessary. The introduction of a new lexical type definitely requires a deep knowledge on the grammar and thus out of the scope of our annotation framework. The other cases can in principle be handled in the current framework, e.g., by a dynamic expansion of the lexicon (i.e., an introduction of a new association between a word and known lexical type), and by a dynamic tuning of the beam size.

To see the significance of the last type of the error, we re-evaluated the annotation results on the Brown sentences after classifying them into: (1) those for which the correct analyses were included in the parser’s chart (in-chart, 65 sentences) and (2) those for which the correct analyses were not in the chart (out-chart, 35 sentences), either because of the pruning effect or the insufficient grammar coverage. The results shown in Table 8 clearly show that there is a large difference in the accuracy of the annotation results between these two cases. Actually, on the in-chart sentences, the parser has returned the correct analysis as the initial solution for over 50% of the sentences, and the annotators saved it without any operations. Thus, we believe it is quite effective to add the above-mentioned functionalities to reduce this type of errors.

6 Conclusion and Future Work

We proposed a new annotation framework for deep grammars by using statistical parsers. From the theoretical point of view, we can achieve significantly high quality HPSG annotations only by CFG annotations, and the products can be useful for the domain adaptation task. On the other hand, preliminary experiments of a manual annotation show some difficulties about CFG annotations for non-experts, especially grammar-specific ones. We hence need to develop some bridging functions reducing such difficulties. One possible strategy is to introduce another representation such as flat CFG than binary CFG. While we adopted CFG interface in our first prototype system, our scheme can be applied to another interface such as dependency as long as there exist some relatedness over syntax or semantics.

References

David Carter. 1997. The treebanker: a tool for supervised training of parsed corpora. In *Workshop On Computational Environments For Grammar Development And Linguistic Engineering*, pages 9–15.

Tadayoshi Hara, Yusuke Miyao, and Jun'ichi Tsujii. 2007. Evaluating impact of re-training a lexical disambiguation model on domain adaptation of an hpsg parser. In *Proceedings of the 10th International Conference on Parsing Technologies*, pages 11–22, Prague, Czech Republic.

Nancy Ide, Collin Baker, Christiane Fellbaum, and Rebecca Passonneau. 2010. The manually annotated sub-corpus: A community resource for and by the people. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 68–73, Uppsala, Sweden, July.

Sadao Kurohashi and Makoto Nagao. 1998. Building a japanese parsed corpus while improving the parsing system. In *Proceedings of the NLPRS*, pages 719–724.

Henry Kučera and W. Nelson Francis. 1967. *Computational Analysis of Present Day American English*. Brown University Press, June.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn treebank: Annotating predicate argument structure. In *Proceedings of the Workshop on Human Language Technology*, pages 114–119.

Takashi Ninomiya, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2007. A log-linear model with an n-gram reference distribution for accurate hpsg parsing. In *Proceedings of the 10th International Conference on Parsing Technologies*, pages 60–68.

Carl Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

Ricardo Sánchez-Sáez, Joan-Andreu Sánchez, and José-Miguel Benedí. 2009. Interactive predictive parsing. In *Proceedings of the 11th International Conference on Parsing Technologies*, pages 222–225.

Ricardo Sánchez-Sáez, Luis A. Leiva, Joan-Andreu Sánchez, and José-Miguel Benedí. 2010. Interactive predictive parsing using a web-based architecture. In *Proceedings of the NAACL HLT 2010 Demonstration Session*, pages 37–40.

Yi Zhang and Valia Kordoni. 2010. Discriminant ranking for efficient treebanking. In *Coling 2010: Posters*, pages 1453–1461, Beijing, China, August. Coling 2010 Organizing Committee.