Adapting a Probabilistic Disambiguation Model of an HPSG Parser to a New Domain

Tadayoshi Hara[1], Yusuke Miyao[1], and Jun'ichi Tsujii[1,2,3]

[1] Department of Computer Science, University of Tokyo,
Hongo 7-3-1, Bunkyo-ku, Tokyo 113-0033, Japan
[2] CREST, JST (Japan Science and Technology Agency),
Honcho, 4-1-8, Kawaguchi-shi, Saitama 332-0012, Japan
[3] School of Informatics, University of Manchester,
POBox 88, Sackville St, Manchester, M60 1QD, UK

Abstract. This paper describes a method of adapting a domain-independent HPSG parser to a biomedical domain. Without modifying the grammar and the probabilistic model of the original HPSG parser, we develop a log-linear model with additional features on a treebank of the biomedical domain. Since the treebank of the target domain is limited, we need to exploit an original disambiguation model that was trained on a larger treebank. Our model incorporates the original model as a reference probabilistic distribution. The experimental results for our model trained with a small amount of a treebank demonstrated an improvement in parsing accuracy.

1 Introduction

Natural language processing (NLP) is being demanded in various fields, such as biomedical research, patent application, and WWW, because an unmanageable amount of information is being published in unstructured data, i.e., natural language texts. To exploit latent information in these, the assistance of NLP technologies is highly required. However, an obstacle is the lack of portability of NLP tools. In general, NLP tools specialized to each domain were developed from scratch, or adapted by considerable human effort. This is because linguistic resources for each domain, such as a treebank, have not been sufficiently developed yet. Since dealing with various kinds of domains is an almost intractable job, sufficient resources can not be expected.

The method presented in this paper is the development of disambiguation models of an HPSG parser by combining a disambiguation model of an original parser with a new model adapting to a new domain. Although the training of a disambiguation model of a parser requires a sufficient amount of a treebank, its construction requires a considerable human effort. Hence, we exploit the original disambiguation model that was trained with a larger, but domain-independent treebank. Since the original disambiguation model contains rich information of general grammatical constraints, we try to use its information in developing a disambiguation model for a new domain.

R. Dale et al. (Eds.): IJCNLP 2005, LNAI 3651, pp. 199–210, 2005.

Our disambiguation model is a log-linear model into which the original disambiguation model is incorporated as a reference distribution. However, we cannot simply estimate this model, because of the problem that has been discussed in studies of the probabilistic modeling of unification-based grammars [1,2]. That is, the exponential explosion of parse candidates assigned by the grammar makes the estimation intractable. The previous studies solved the problem by applying a dynamic programming algorithm to a packed representation of parse trees. In this paper, we borrow their idea, and define reference distribution on a packed structure. With this method, the log-linear model with a reference distribution can be estimated by using dynamic programming.

In the experiments, we used an HPSG parser originally trained with the Penn Treebank [3], and evaluated a disambiguation model trained with the GENIA treebank [4], which consisted of abstracts of biomedical papers. First, we measured the accuracy of parsing and the time required for parameter estimation. For comparison, we also examined other possible models other than our disambiguation model. Next, we varied the size of a training corpus in order to evaluate the size sufficient for domain adaptation. Then, we varied feature sets used for training and examined the parsing accuracy. Finally, we compared the errors in the parsing results of our model with those of the original parser.

In Section 2, we introduce the disambiguation model of an HPSG parser. In Section 3, we describe a method of adopting reference distribution for adapting a probabilistic disambiguation model to a new domain. In Section 4, we examine our method through experiments on the GENIA treebank.

2 An HPSG Parser

The HPSG parser used in this study is *Enju* [5]. The grammar of Enju was extracted from the Penn Treebank [3], which consisted of sentences collected from The Wall Street Journal [6]. The disambiguation model of Enju was trained on the same treebank. This means that the parser has been adapted to The Wall Street Journal, and would be difficult to apply to other domains such as biomedical papers that include different distribution of words and their constraints.

In this study, we attempted the adaptation of a probabilistic disambiguation model by fixing the grammar and the disambiguation model of the original parser. The disambiguation model of Enju is based on a feature forest model [2], which is a maximum entropy model [7] on packed forest structure. The probability, $p_E(t|s)$, of producing the parse result t for a given sentence s is defined as

$$p_E(t|s) = \frac{1}{Z_s} \exp\left(\sum_i \lambda_i f_i(t,s)\right)$$

$$Z_s = \sum_{t' \in T(s)} \exp\left(\sum_i \lambda_i f_i(t',s)\right),$$

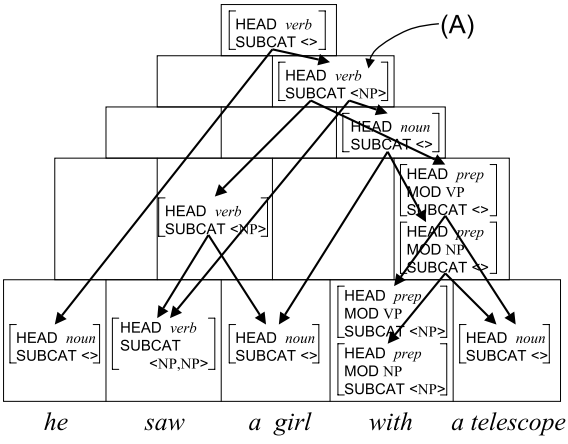

Fig. 1. Chart for parsing *"he saw a girl with a telescope"*

where $T(s)$ is the set of parse candidates assigned to s. The feature function $f_i(t, s)$ represents the characteristics of t and s, while the corresponding model parameter λ_i is its weight. Model parameters were estimated so as to maximize the log-likelihood of the training data.

Estimation of the above model requires a set of training pairs $\langle t_s, T(s)\rangle$, where t_s is the correct parse for the sentence s. While t_s is provided by a treebank, $T(s)$ is computed by parsing each s in the treebank. However, the simple enumeration of $T(s)$ is impractical because the size of $T(s)$ is exponential to the length of s.

To avoid an exponential explosion, Enju represented $T(s)$ in a packed form of HPSG parse trees [5]. In chart parsing, partial parse candidates are stored in a *chart*, in which phrasal signs are identified and packed into an equivalence class if they are determined to be equivalent and dominate the same word sequence. A set of parse trees is then represented as a set of relations among equivalence classes. Figure 1 shows a chart for parsing *"he saw a girl with a telescope"*, where the modifiee (*"saw"* or *"girl"*) of *"with"* is ambiguous. Each feature structure expresses an equivalence class, and the arrows represent immediate-dominance relations. The phrase, *"saw a girl with a telescope"*, has two ambiguous subtrees (A in the figure). Since the signs of the top-most nodes are equivalent, they are packed into the same equivalence class. The ambiguity is represented as two pairs of arrows that come out of the node.

A packed chart can be interpreted as an instance of a feature forest [2]. A feature forest represents a set of exponentially-many trees in an "and/or" graph of a tractable size. A feature forest is formally defined as a tuple $\langle C, D, R, \gamma, \delta\rangle$, where C is a set of conjunctive nodes, D is a set of disjunctive nodes, $R \subseteq C$ is a set of root nodes[1], $\gamma : D \rightarrow 2^C$ is a conjunctive daughter function, and $\delta : C \rightarrow 2^D$ is a disjunctive daughter function.

[1] For the ease of explanation, the definition of root node is slightly different from the original.

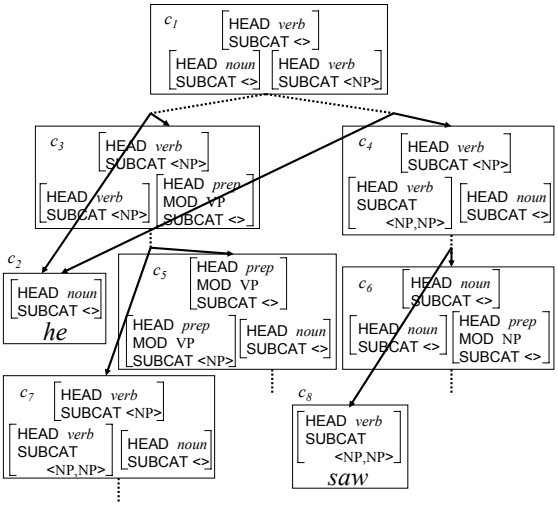

Fig. 2. Packed representation of HPSG parse trees in Figure 1

Figure 2 shows (a part of) the HPSG parse trees in Figure 1 represented as a feature forest. Square boxes are conjunctive nodes, dotted lines express a disjunctive daughter function, and solid arrows represent a conjunctive daughter function.

Based on the definition, parse tree t of sentence s can be represented as the set of conjunctive nodes in the feature forest. The probability $p_E(t|s)$ is then redefined as

$$p_E(t|s) = \frac{1}{Z_s} \exp\left(\sum_{c \in t} \sum_i \lambda_i f_i(c)\right)$$

$$Z_s = \sum_{t' \in T(s)} \exp\left(\sum_{c \in t'} \sum_i \lambda_i f_i(c)\right),$$

where $f_i(c)$ are alternative feature functions assigned to conjunctive nodes $c \in C$. By using this redefined probability, a dynamic programming algorithm can be applied to estimate $p(t|T(s))$ without unpacking the packed chart [2].

Feature functions in feature forest models are designed to capture the characteristics of a conjunctive node. In HPSG parsing, it corresponds to a tuple of a mother and its daughters. Enju uses features that are combinations of the atomic features listed in Table 1. The following combinations are used for representing the characteristics of the binary/unary rule applications.

$$f_{\text{binary}} = \left\langle \begin{array}{l} \text{RULE,DIST,COMMA,} \\ \text{SPAN}_h, \text{SYM}_h, \text{WORD}_h, \text{POS}_h, \text{LE}_h, \\ \text{SPAN}_n, \text{SYM}_n, \text{WORD}_n, \text{POS}_n, \text{LE}_n \end{array} \right\rangle$$

$$f_{\text{unary}} = \langle \text{RULE,SYM,WORD,POS,LE} \rangle$$

where suffix h and n means a head daughter and a non-head daughter, respectively.

Table 1. Templates of atomic features

RULE	the name of the applied schema
DIST	the distance between the head words of the daughters
COMMA	whether a comma exists between daughters and/or inside of daughter phrases
SPAN	the number of words dominated by the phrase
SYM	the symbol of the phrasal category (e.g. NP, VP)
WORD	the surface form of the head word
POS	the part-of-speech of the head word
LE	the lexical entry assigned to the head word

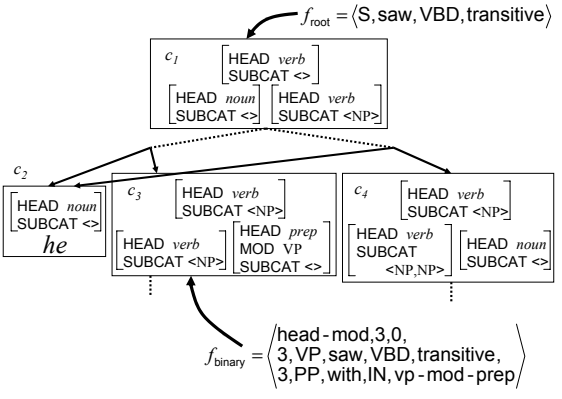

Fig. 3. Example features

In addition, the following feature is used for expressing the condition of the root node of the parse tree.

$$f_{\text{root}} = \langle \text{SYM,WORD,POS,LE} \rangle$$

Figure 3 shows example features: f_{root} is the feature for the root node, in which the phrase symbol is S and the surface form, part-of-speech, and lexical entry of the lexical head are *"saw"*, VBD, and a transitive verb, respectively. The f_{binary} is the feature for the binary rule application to *"saw a girl"* and *"with a telescope"*, in which the applied schema is the Head-Modifier Schema, the head daughter is VP headed by *"saw"*, and the non-head daughter is PP headed by *"with"*, whose part-of-speech is IN and the lexical entry is a VP-modifying preposition.

3 Re-training of Disambiguation Models

The method of domain adaptation is to develop a new maximum entropy model with incorporating an original model as a reference probabilistic distribution. The idea of adaptation using a reference distribution has already been presented

in several studies [8,9]. When we have a reference probabilistic model $p_0(t|s)$ and are making a new model $p_M(t|s)$, the probability is defined as

$$p_M(t|s) = \frac{1}{Z'_s} p_0(t|s) \exp\left(\sum_j \rho_j g_j(t', s)\right)$$

$$\text{where} \quad Z'_s = \sum_{t' \in T(s)} p_0(t'|s) \exp\left(\sum_j \rho_j g_j(t', s)\right).$$

Model parameters, ρ_j, are estimated so as to maximize the likelihood of the training data as in ordinary maximum entropy models. The maximization of the likelihood with the above model is equivalent to finding the model p_M that is closest to the reference probability p_0 in terms of the Kullback-Leibler distance.

However, we cannot simply apply the above method to our task because the parameter estimation requires the computation of the above probability for all parse candidates $T(s)$. As discussed in Section 2, the size of $T(s)$ is exponentially related to the length of s. This imposes a new problem, that is, we need to enumerate $p_0(t|s)$ for all candidate parses. Obviously, this is intractable.

Since Enju represented a probabilistic disambiguation model in a packed forest structure, we exploit that structure to represent our probabilistic model. That is, we redefine p_M with feature functions g_j on conjunctive nodes as

$$p_M(t|s) = \frac{1}{Z'_s} p_0(t|s) \exp\left(\sum_{c \in t} \sum_j \rho_j g_j(c)\right)$$

$$\text{where} \quad Z'_s = \sum_{t' \in T(s)} p_0(t|s) \exp\left(\sum_{c \in t'} \sum_j \rho_j g_j(c)\right).$$

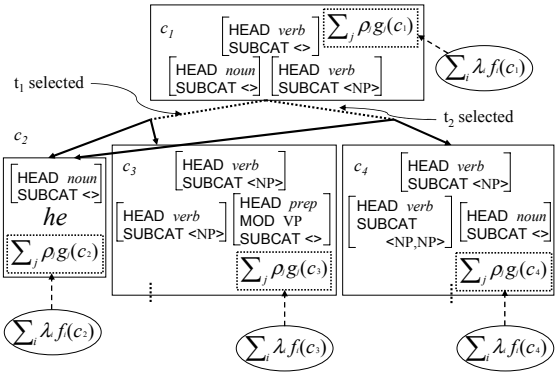

Fig. 4. Example of importing a reference distribution into each conjunctive node

As described in Section 2, the original model, $p_E(t|s)$, is expressed in a packed structure as

$$p_E(t|s) = \frac{1}{Z_s} \exp\left(\sum_{c\in t}\sum_i \lambda_i f_i(c)\right)$$

$$\text{where} \quad Z_s = \sum_{t'\in T(s)} \exp\left(\sum_{c\in t}\sum_i \lambda_i f_i(c)\right).$$

Then, $p_0(t|s)$ is substituted by $p_E(t|s)$, and $p_M(t|s)$ is formulated as

$$p_M(t|s) = \frac{1}{Z'_s}\left\{\frac{1}{Z_s}\exp\left(\sum_{c\in t}\sum_i \lambda_i f_i(c)\right)\right\}\exp\left(\sum_{c\in t}\sum_j \rho_j g_j(c)\right)$$

$$= \frac{1}{Z'_s \cdot Z_s}\exp\left(\sum_{c\in t}\sum_i \lambda_i f_i(c) + \sum_{c\in t}\sum_j \rho_j g_j(c)\right)$$

$$= \frac{1}{Z''_s}\exp\left\{\sum_{c\in t}\left(\sum_i \lambda_i f_i(c) + \sum_j \rho_j g_j(c)\right)\right\}$$

$$\text{where} \quad Z''_s = Z_s \cdot Z'_s = \sum_{t\in T(s)} \exp\left\{\sum_{c\in t}\left(\sum_i \lambda_i f_i(c) + \sum_j \rho_j g_j(c)\right)\right\}.$$

With this form of $p_M(t|s)$, a dynamic programing algorithm can be applied. For example, we show how to obtain probabilities of parse trees in the case of Figure 4. For ease, we assume that there are only two disjunctive daughters (dotted lines) that are of the top conjunctive node. The left disjunctive node introduces a parse tree t_1 that consists of conjunctive nodes $\{c_1, c_2, c_3, \dots\}$, and the right one, t_2 that consists of $\{c_1, c_2, c_4, \dots\}$. To each conjunctive node c_k, a weight from the reference distribution $\sum_i \lambda_i f_i(c_k)$ is assigned. Probability $p_M(t_1|s)$ and $p_M(t_2|s)$ are then given as

$$p_M(t_1|s) = \frac{1}{Z''_s}\exp\left\{\left(\sum_i \lambda_i f_i(c_1) + \sum_j \rho_j g_j(c_1)\right) + \left(\sum_i \lambda_i f_i(c_2) + \sum_j \rho_j g_j(c_2)\right) + \left(\sum_i \lambda_i f_i(c_3) + \sum_j \rho_j g_j(c_3)\right) + \cdots\right\}$$

$$p_M(t_2|s) = \frac{1}{Z''_s}\exp\left\{\left(\sum_i \lambda_i f_i(c_1) + \sum_j \rho_j g_j(c_1)\right) + \left(\sum_i \lambda_i f_i(c_2) + \sum_j \rho_j g_j(c_2)\right) + \left(\sum_i \lambda_i f_i(c_4) + \sum_j \rho_j g_j(c_4)\right) + \cdots\right\}.$$

4 Experiments

We implemented the method described in Section 3. The original parser, Enju, was developed on Section 02-21 of the Penn Treebank (39,832 sentences)[5]. For the training of our model, we used the GENIA treebank [4], which consisted of 500 abstracts (4,446 sentences) extracted from MEDLINE. We divided the GENIA treebank into three sets of 400, 50, and 50 abstracts (3,524, 455, and 467 sentences), and these sets were used respectively as training, development, and final evaluation data. The method of Gaussian MAP estimation [10] was used for smoothing.

The meta parameter σ of the Gaussian distribution was determined so as to maximize the accuracy on the development set. In the following experiments, we measured the accuracy of predicate-argument dependencies on the evaluation set. The measure is labeled precision/recall (LP/LR), which is the same measure as previous work [11,5] that evaluated the accuracy of lexicalized grammars on the Penn Treebank.

First, we measured the accuracy of parsing and the time required for parameter estimation. Table 2 compares the results of the following estimation methods.

Table 2. Accuracy and time cost for various estimation methods

	F-score		Training	Parsing time (sec.)	
	GENIA Corpus	Penn Treebank	time (sec.)	GENIA Corpus	Penn Treebank
Our method	86.87	86.81	2,278	611	3,165
Combined	86.32	86.09	29,421	424	2,757
GENIA only	85.72	42.49	1,694	332	8,183
Original model	85.10	87.16	137,038	515	2,554

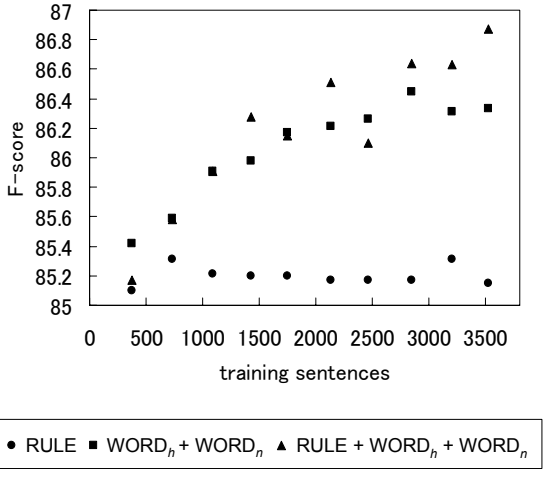

Fig. 5. Corpus size vs. Accuracy

Table 3. Accuracy with atomic feature templates

Features	LP	LR	F-score	diff.
RULE	85.42	84.87	85.15	+0.05
DIST	85.29	84.77	85.03	−0.07
COMMA	85.45	84.86	85.15	+0.05
SPAN_h+SPAN_n	85.58	85.02	85.30	+0.20
SYMBOL_h+SYMBOL_n	85.01	84.56	84.78	−0.32
WORD_h+WORD_n	86.59	86.07	86.33	+1.23
WORD_h	85.48	84.98	85.23	+0.13
WORD_n	85.44	84.64	85.04	−0.06
POS_h+POS_n	85.23	84.77	85.00	−0.10
LE_h+LE_n	85.42	85.06	85.24	+0.14
None	85.39	84.82	85.10	

Table 4. Accuracy with the combination of RULE and other features

Features	LP	LR	F-score	diff.
RULE+DIST	85.41	84.85	85.13	+0.03
RULE+COMMA	85.92	85.15	85.53	+0.43
RULE+SPAN_h+SPAN_n	85.33	84.82	85.07	−0.03
RULE+SYMBOL_h+SYMBOL_n	85.43	85.00	85.21	+0.11
RULE+WORD_h+WORD_n	87.12	86.62	86.87	+1.77
RULE + WORD_h	85.74	84.94	85.34	+0.24
RULE + WORD_n	85.10	84.60	84.85	−0.25
RULE+POS_h+POS_n	85.51	85.08	85.29	+0.19
RULE+LE_h+LE_n	85.48	85.08	85.28	+0.18
None	85.39	84.82	85.10	

Our method: training with our method
Combined: training Enju model with the training corpus replaced by the combination of the GENIA corpus and the Penn Treebank
GENIA only: training Enju model with the training corpus replaced by the GENIA corpus only
Original Model: training an original Enju model

The table shows the accuracy and the parsing time for the GENIA corpus and the Penn Treebank Section 23, and also shows the time required for the training of the model. The additional feature used in our method was RULE+WORD_h+WORD_n, which will be explained later. In the "Combined" method, we could not train the model with the original training parameters ($n = 20$, $\epsilon = 0.98$ in [5]) because the estimator ran out of memory. Hence, we reduced the parameters to $n = 10$, $\epsilon = 0.95$.

For the GENIA corpus, our model gave the higher accuracy than the original model and the other estimation methods, while for the Penn Treebank, our model gave a little lower accuracy than the original model. This result indicates that our model was more adapted to the specific domain. The "GENIA only"

Table 5. Accuracy with the combination of WORD and another feature

Features	LP	LR	F-score	diff.
WORD_h+WORD_n+RULE	87.12	86.62	86.87	+1.77
WORD_h+WORD_n+DIST	86.41	85.86	86.14	+1.04
WORD_h+WORD_n+COMMA	86.91	86.38	86.64	+1.54
WORD_h+WORD_n+SPAN_h+SPAN_n	85.77	85.22	85.49	+0.39
WORD_h+WORD_n+SYMBOL_h+SYMBOL_n	86.58	85.70	86.14	+1.04
WORD_h+WORD_n+POS_h+POS_n	86.53	85.99	86.26	+1.16
WORD_h+WORD_n+LE_h+LE_n	86.16	85.68	85.92	+0.82
None	85.39	84.82	85.10	

Table 6. Errors in our model and Enju

	Total errors	Common errors	Errors not in the other model
Our model	1179	1050	129
Original model	1338	1050	288

method gave significantly lower accuracy. We expect that the method clearly lacked the amount of the training corpus for obtaining generic grammatical information.

The "Combined" method achieved the accuracy close to our method. However, it is notable that our method took much less time for the training of the model since ours did not need to handle the Penn Treebank. Instead, our method exploited the original model of Enju, which was trained on the Penn Treebank, and this resulted in much less cost of training.

Next, we changed the size of the GENIA treebank for training: 40, 80, 120, 160, 200, 240, 280, 320, 360, and 400 abstracts. Figure 5 shows the accuracy when the size of the training data was changed. We can say that, for those feature sets giving remarkable accuracy in the experiments, the accuracy edged upwards with the size of the training corpus, and the trend does not seem to converge even if more than 400 abstracts exist. If we choose more complex feature sets for higher accuracy, data sparseness will occur and an even larger corpus will be needed. These findings indicate that we can further improve the accuracy by using a larger treebank and a proper feature set.

Table 3 shows the accuracy of models with only atomic feature templates. The bottom of the table gives the accuracy attained by the original parser. When we focus on the WORD features, we can see the combination of WORD_h and WORD_n improved the accuracy significantly, although each of the features by itself did not improve so much. DIST, SYMBOL, and POS feature templates lowered the accuracy. The other feature templates improved the accuracy, though not as well as the WORD templates.

Table 4 shows that the RULE feature combined with one or more other features often gave a little higher accuracy than the RULE feature gave by itself, though not as well as the WORD features.

Table 5 shows that the WORD features combined with one or more other features gave remarkable improvement to the accuracy as a whole. RULE and COMMA features gave even higher accuracy than with only the WORD features. Our results revealed that the WORD features were crucial for the adaptation to the biomedical domain. We expect that this was because the biomedical domain had a different distribution of words, while more generic grammatical constraints were not significantly different from other domains.

Table 6 shows the comparison of the number of errors of our model with those of the original model in parsing the GENIA corpus. Though our model gave less errors than the original model, our model introduced a certain amount of new errors. In future work, we need to investigate manually those errors to find more suitable feature templates without losing the information in the original model.

5 Conclusions

We have presented a method of adapting a domain-independent HPSG parser to a biomedical domain. Since the treebank of the new domain was limited, we exploited an original disambiguation model. The new model was trained on a biomedical treebank, and was combined with the original model by using it as a reference distribution of a log-linear model. The experimental results demonstrated our new model was adapted to the target domain, and was superior to other adaptation methods in accuracy and the cost of training time. With our model, the parsing accuracy for the target domain improved by 1.77 point with the treebank of 3,524 sentences. Since the accuracy did not seem to saturate, we will further improve the accuracy by increasing the size of the domain-dependent treebank. In addition, the experimental results showed that the WORD feature significantly contributed to the accuracy improvement.

We examined only a few feature templates, and we must search for further more feature templates. Not only the new combinations of the atomic features but also new types of features, which may be domain-dependent such as named entities, will be possible.

References

1. Geman, S., Johnson, M.: Dynamic programming for parsing and estimation of stochastic unification-based grammars. In: Proc. 40th ACL. (2002)
2. Miyao, Y., Tsujii, J.: Maximum entropy estimation for feature forests. In: Proc. HLT 2002. (2002)
3. Marcus, M., Kim, G., Marcinkiewicz, M.A., MacIntyre, R., Bies, A., Ferguson, M., Katz, K., Schasberger, B.: The Penn Treebank: Annotating predicate argument structure. In: ARPA Human Language Technology Workshop. (1994)
4. Kim, J.D., Ohta, T., Teteisi, Y., Tsujii, J.: Genia corpus - a semantically annotated corpus for bio-textmining. Bioinformatics **19** (2003) i180–i182
5. Miyao, Y., Tsujii, J.: Probabilistic disambiguation models for wide-coverage HPSG parsing. In: Proc. ACL 2005. (2005)

6. Miyao, Y., Ninomiya, T., Tsujii, J.: Corpus-oriented grammar development for acquiring a Head-driven Phrase Structure Grammar from the Penn Treebank. In: Proc. IJCNLP-04. (2004)
7. Berger, A.L., Pietra, S.A.D., Pietra, V.J.D.: A maximum entropy approach to natural language processing. Computational Linguistics **22** (1996) 39–71
8. Jelinek, F.: Statistical Methods for Speech Recognition. The MIT Press (1998)
9. Johnson, M., Riezler, S.: Exploiting auxiliary distributions in stochastic unification-based grammars. In: Proc. 1st NAACL. (2000)
10. Chen, S., Rosenfeld, R.: A gaussian prior for smoothing maximum entropy models. Technical Report CMUCS-99-108, Carnegie Mellon University (1999)
11. Clark, S., Curran, J.R.: Parsing the WSJ using CCG and log-linear models. In: Proc. 42nd ACL. (2004)