Japanese Dependency Parsing Using a Tournament Model

Masakazu Iwatate and **Masayuki Asahara** and **Yuji Matsumoto**
Nara Institute of Science and Technology, Japan
8916-5, Takayama, Ikoma, Nara, Japan, 630-0192
{masakazu-i, masayu-a, matsu}@is.naist.jp

Abstract

In Japanese dependency parsing, Kudo's relative preference-based method (Kudo and Matsumoto, 2005) outperforms both deterministic and probabilistic CKY-based parsing methods. In Kudo's method, for each dependent word (or chunk) a log-linear model estimates relative preference of all other candidate words (or chunks) for being as its head. This cannot be considered in the deterministic parsing methods. We propose an algorithm based on a tournament model, in which the relative preferences are directly modeled by one-on-one games in a step-ladder tournament. In an evaluation experiment with *Kyoto Text Corpus Version 4.0*, the proposed method outperforms previous approaches, including the relative preference-based method.

1 Introduction

The shared tasks of multi-lingual dependency parsing took place at CoNLL-2006 (Buchholz and Marsi, 2006) and CoNLL-2007 (Nivre et al., 2007). Many language-independent parsing algorithms were proposed there. The algorithms need to adapt to various dependency structure constraints according to target languages: projective vs. non-projective, head-initial vs. head-final, and single-rooted vs. multi-rooted. Eisner (1996) proposed a CKY-like $O(n^3)$ algorithm. Yamada and Matsumoto (2003) proposed a shift-reduce-like $O(n^2)$ deterministic algorithm. Nivre et al. (2003; 2004) also proposed a shift-reduce-like $O(n)$ deterministic algorithm for projective languages. The model is enhanced for non-projective languages by Nivre and Nilsson (2005). McDonald et al. (2005) proposed a method based on search of maximum spanning trees employing the Chu-Liu-Edmonds algorithm (hereafter "**CLE algorithm**") (Chu and Liu, 1965; Edmonds, 1967). Most Japanese dependency parsers are based on *bunsetsu units*, which are similar concept to English base phrases. The constraints in Japanese dependency structure are stronger than those in other languages. Japanese dependency structures have the following constraints: head-final, single-head, single-rooted, connected, acyclic and projective. Figure 1 shows examples of Japanese sentences and their dependency structures. Each box represents a *bunsetsu*. A dependency relation is represented by an edge from a dependent to its head. Though sentence (a) is similar to sentence (b), the syntactic structures of these two are different, especially because "*kare-wa*" directly depends on "*yomanai*" in (b) but not in (a).

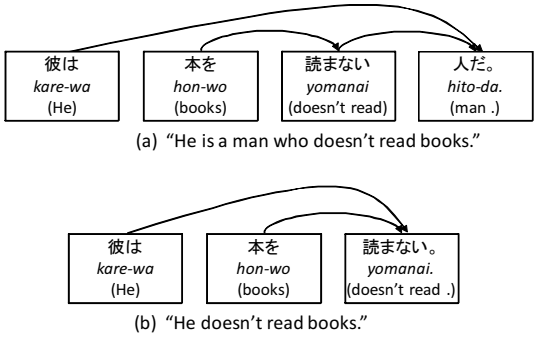

Figure 1: Examples of Japanese sentences.

In dependency parsing of Japanese, deterministic algorithms outperform probabilistic CKY methods. Kudo and Matsumoto (2002) applied the

Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008), pages 361–368
Manchester, August 2008

cascaded chunking algorithm (hereafter "**CC algorithm**") to Japanese dependency parsing. Yamada's method (Yamada and Matsumoto, 2003) employed a similar algorithm. Sassano (2004) proposed a linear-order shift-reduce-like algorithm (hereafter "**SR algorithm**"), which is similar to Nivre's algorithm (Nivre, 2003). These deterministic algorithms are biased to select nearer candidate heads since they examine the candidates sequentially, and once they find a plausible one they never consider further candidates.

We experimented the CLE algorithm with Japanese dependency parsing, and found that the CLE algorithm is comparable to or in some cases poorer than the deterministic algorithms in our experiments. Actually, the CLE algorithm is not suitable for some of the constraints in Japanese dependency structures: head-final and projective. First, head-final means that dependency relation always goes from left to right. Second, since the CLE algorithm may produce non-projective dependency trees, we need to conduct projectivity check in the algorithm.

Kudo and Matsumoto (2005) proposed a relative preference-based method (hereafter "**relative preference method**"). They defined the parsing algorithm as series of selection steps of the most likely head for each *bunsetsu* out of all candidates. The method has so far achieved the highest accuracy in the experiments with *Kyoto Text Corpus Version 3.0* data [1], since other deterministic methods do not consider relative preference among candidate heads but solely consider whether the focused-on pair of *bunsetsu's* is in a dependency relation or not.

We propose a model that takes a *bunsetsu* and two candidate heads into consideration and selects the better candidate head out of those two. This step is repeated in a step ladder tournament to get the best candidate head (hereafter we call this model as a "**tournament model**"). The tournament model was first introduced by Iida et al. (2003) for coreference resolution. We applied this model to selecting the most plausible candidate head for each *bunsetsu* except for the sentence final one.

Section 2 describes the tournament model comparing with previous research. Section 3 describes how the tournament model is applied to Japanese dependency parsing. Section 4 shows the results of evaluation experiments. Section 5 shows our current and future work, and Section 6 gives conclusions of this research.

[1]Note: Sassano's SR algorithm is the highest by experiment with the smaller data *Kyoto Text Corpus Version 2.0* Relative preference method and SR algorithm are not compared directly with the same data.

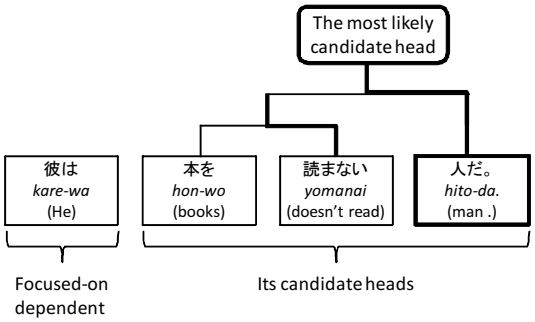

Figure 2: Example of a tournament.

2 Tournament Model

The tournament model was first introduced by Iida et al. (2003) for coreference resolution. The model chooses the most likely candidate in a step-ladder tournament, that is a sequence of one-on-one games between candidate referents for a given anaphoric expression. In each game, the winner is chosen by a binary classifier such as SVMs.

We applied the tournament model to Japanese dependency parsing taking into consideration Japanese constraints. The projective constraint is easily met. When selecting candidate heads for the focused-on dependent, we only consider those candidates that introduce no crossing dependency.

Figure 2 illustrates a tournament. The focused-on dependent *bunsetsu* is "*kare-wa*", and the candidate heads are the three *bunsetsu's* on the right-hand side: "*hon-wo*", "*yomanai*" and "*hito-da*". The first game is "*hon-wo*" vs. "*yomanai*". Then the next game is the winner of the first game vs. "*hito-da*". The winner of the second game (i.e., "*hito-da*") is chosen as the most likely candidate of the dependent, "*kare-wa*".

In the tournament model, the most likely head of a given bunsetsu is determined by a series of one-on-one games in a tournament. Below, we present the advantages of the tournament model by comparison with the previous methods.

2.1 Scope of Feature Views

The CC algorithm and SR algorithm consider only a pair of *bunsetsu's* – a dependent and its candidate

head – in the parsing action determination (hereafter "**2-tuple model**"). The same 2-tuple may or may not have a dependency relation when they appear in different context. For example, both (a) and (b) in Figure 1 include the two *bunsetsu's*, "*kare-wa*" and "*yomanai*"; in (b) they have a dependency relation, but not in (a). The 2-tuple models and relative preference method cannot discriminate between these two patterns without considering contextual features [2]. The tournament model can be regarded as a "**3-tuple model**," which considers three *bunsetsu's* – a dependent and two candidate heads. The discriminative performance of the 3-tuple model is greater than the 2-tuple models, since it directly compares two candidate heads and selects the one that is more plausible than the other candidate. Consider Figure 1 again. In (a), "*kare-wa*" does not depend on "*yomanai*" because there is another *bunsetsu* "*hito-da*" which is a more plausible head. 2-tuple models may use this information as a contextual feature, but the effect is indirect. On the other hand, the tournament model directly compares these candidates and always selects the better one. The situation becomes crucial when the true head appears outside of the context window of the current candidate. 2-tuple models have to select the head without consulting such information. The advantage of the tournament model is its capability of deferring the decision by always keeping the current best candidate head. On the other hand, a disadvantage of the tournament model is its space and time complexity. The size of features is larger since they come from three *bunsetsu's*. The size of training instances is also larger.

2.2 Relative Position in a Sentence

We name the two candidate heads in the 3-tuple model as "the nearer candidate head" and "the farther candidate head." The dependent, the nearer candidate head and the farther candidate head appear in this order in Japanese sentences. The order defines the relative position of the contextual features. The distance between the dependent and a candidate head is another feature to represent the relative position. In previous research, the distance has been represented by feature buckets, such as 1, 2-5, or 6+. While for some dependents and their heads whether the distance is 1 or not is important, absolute distance is not so important since Japanese is a free-order language. Relative positions are more informative since some dependents tend to appear closer to other dependents, such as objects that tend to appear closer to predicates compared with other complements. The tournament model represents both the distance and relative position as features.

The deterministic algorithms are biased to select nearer candidate heads. As most dependent and head pairs appear within a close window, this tendency does not cause many errors; deterministic algorithms are weak at finding correct heads that appear in a long distance as pointed out in Kudo and Matsumoto (2005).

2.3 Relative Preferences

What the dependency parsers try to learn is relative preference of *bunsetsu* dependency, i.e., how a dependent selects its head among others. The relative preference method (Kudo and Matsumoto, 2005) learns the relative preferences among the candidate heads by a discriminative framework. The relative preferences are learned with the log-linear model so as to give larger probability to the correct dependent-head pair over any other candidates. McDonald's method (2005) with the CLE algorithm learns the relative preferences by a perceptron algorithm – MIRA (Crammer and Singer, 2003), so that the correct dependent-head link receives a higher score. The tournament model learns which candidate is more likely to be the head between two candidates in a one-on-one game in a tournament. Therefore, all of those parsing algorithms try to learn the way to give the highest preference to the correct dependent-head pair among all possibilities though in different settings.

While the relative preference method and McDonald's method consider all candidate heads independently in a discriminative model, the tournament model evaluates which candidate is more likely to be the head between the latest winner and the new candidate. The latest winner has already defeated all of the preceding candidates. If the new candidate beats the latest winner, it becomes the new winner, meaning that it is the most preferred candidate among others so far considered. Through this way of comparison with the runner-up candidates, the tournament model uses richer information in learning relative preferences than the models in which all candidates are independently considered.

[2] Contextual features are features neither in the dependent nor in the candidate head(s).

```
// N: # of bunsetsu's in input sentence
// true_head[j]: bunsetsu j's head at
//                  training data
// gen(j,i1,i2,LEFT): generate
//   an example where bunsetsu j is
//   dependent of i1
// gen(j,i1,i2,RIGHT): generate
//   an example where bunsetsu j is
//   dependent of i2

for j = 1 to N-1 do
  h = true_head[j];
  for i = j+1 to h-1 do
            gen(j,i,h,RIGHT);
  for i = h+1 to N do
            gen(j,h,i,LEFT);
end-for;
```

Figure 3: Pseudo code of training example generation procedure.

```
// N: # of bunsetsu's in
//        input sentence
// head[]: (analyzed-) head of bunsetsu
// classify(j,i1,i2): ask SVM
//   which candidate  (i1 or i2) is
//   more likely for head of j.
//   return LEFT if i1 wins.
//   return RIGHT if i2 wins.

head[] = {2,3,...,N-1,N,EOS};
for j = N-1 downto 1 do
  h = j+1;
  i = head[h];
  while i != EOS do
    if classify(j,h,i)==RIGHT
         then h = i;
    i = head[i];
  end-while;
  head[j] = h;
end-for;
```

Figure 4: Pseudo code of parsing algorithm.

3 Proposed Algorithm

3.1 Training Example Generation Algorithm

As shown in Figure 3, for each dependent, we generate pairs of the correct head and all other candidate heads. On the example generation, the procedure does not take into account the projective constraint; all *bunsetsu's* on the right-hand side of the focused-on dependent are candidate heads.

Table 1 shows all examples generated from two sentences shown in Figure 1. 2-tuple models generate training examples formed as (dependent, candidate). So, from the sentences of Figure 1, it generates opposite classes to the pair (*kare-wa*, *hito-da*). On the other hand, the examples generated by the tournament model do not contain such inconsistency.

3.2 Parsing Algorithm

The tournament model has quite wide freeness in the parsing steps. We introduce one of the tournament algorithms, in which the dependents are picked from right to left; and the games of the tournament are performed from left to right. This parsing algorithm takes into account the projective and head-final constraints.

This algorithm is shown in Figure 4. The overall parsing process moves from right to left. On selecting the head for a dependent all of the *bunsetsu*'s to the right of the dependent have already been decided. In Figure 4, the array "`head`" stores the parsed results and ensures that only non-crossing candidate heads are taken into consideration.

Note that the structure of the tournament has little effect on the results (< 0.1) in our preliminary experiments. We tried 2×2 options: the dependents are picked from right to left or from left to right; and the games of the tournament are performed from right to left or from left to right. We choose the most natural combination for Japanese dependency parsing, which is easy to implement.

4 Experiment

4.1 Settings

We implemented the tournament model, the CC algorithm (Kudo and Matsumoto, 2002), SR algorithm (Sassano, 2004) and CLE algorithm (McDonald et al., 2005) with SVM classifiers. We evaluated dependency accuracy and sentence accuracy using *Kyoto Text Corpus Version 4.0*, which is composed by newspaper articles. Dependency accuracy is the percentage of correct dependencies out of all dependency relations. Sentence accuracy is the percentage of sentences in which all dependencies are determined correctly. Dependency accuracy is calculated excluding the rightmost *bunsetsu* of each sentence. [3] Sentences that consist of one *bunsetsu* are not used in our experiments.

We use January 1st to 8th (7,587 sentences) for the training data. We use January 9th (1,213 sentences), 10th (1,479 sentences) and 15th (1,179 sentences) for the test data. We use TinySVM [4] as a binary classifier. Cubic polynomial kernel is

[3] Most research such as Kudo's (2005) uses this criteria.

[4] `http://chasen.org/~taku/software/TinySVM/`

Sentence	Focused-on dependent	Left(Nearer) candidate	Right(Farther) candidate	Class label
(a)	*kare-wa*	*hon-wo*	*hito-da.*	RIGHT
(a)	*kare-wa*	*yomanai*	*hito-da.*	RIGHT
(a)	*hon-wo*	*yomanai*	*hito-da.*	LEFT
(b)	*kare-wa*	*hon-wo*	*yomanai.*	RIGHT

Table 1: Generated examples from sentences in Figure 1.

used for the kernel function. Cost of constraint violation is 1.0. These SVM settings are the same as previous research (Kudo and Matsumoto, 2002; Sassano, 2004). All experiments were performed on Dual Core Xeon 3GHz x 2 Linux machines.

4.2 Features

Here we describe features used in our experiments. Note that for the tournament model, features corresponding to candidates are created for each of the nearer and farther candidates. We define the *information* of a word as the following features: lexical forms, coarse-grained POS tags, full POS tags and inflected forms. We also define the *information* of a *bunsetsu* as word *information* for each of *syuji* and *gokei*. *Syuji* is the head content word of the *bunsetsu*, defined as the rightmost content word. *Gokei* is the representative function word of the *bunsetsu*, defined as the rightmost functional word.

Existence of punctuations or brackets, whether the *bunsetsu* is the first bunsetsu in the sentence, and whether it is the final bunsetsu in the sentence are also members of *information* of a *bunsetsu*.

Standard features are the following: *Information* of the dependent and the candidate heads, distance between the dependent and the candidate heads (1, 2-5 or 6+ *bunsetsu's*), all punctuations, brackets and all particles between the dependent and the candidate heads.

Additional features are the following: All case particles in the dependent and the candidate heads, *information* of the leftmost word in the candidate heads, and the lexical form of the neighboring *bunsetsu* to the right of the candidate heads.

Case particle features are the following: All case particles appearing in the candidates' dependent. These features are intended to take into consideration the correlation between the case particles in the dependent of a head. When the head is a verb, it has a similar effect of learning case frame information.

Standard and additional features are introduced by Sassano (2004). The case particle feature is newly introduced in this paper. Features corresponding to the already-determined dependency relation are called *dynamic* features, and the other contextual features are called *static* features. Standard and additional features are static features, and case particle features are dynamic features. Whether a dynamic feature is available for a parsing algorithm depends on the parsing order of the algorithm.

4.3 Parsing Accuracy

The parsing accuracies of our model and previous models are summarized in Table 2. Note that, since the CLE algorithm is non-deterministic and dynamic features are not available for this algorithm, we use only a *standard and additional* feature set instead of an *all* feature set. By McNemar test ($p < 0.01$) on the dependency accuracy, the tournament model significantly outperforms most of other methods except for the SR algorithm on January 10th data with all features ($p = 0.083$) and the CC algorithm on January 10th data with all features ($p = 0.099$). The difference between the tournament models with all features and with the standard feature only is significant except for on January 9th data ($p = 0.25$).

The highest dependency accuracy reported for January 9th of *Kyoto Text Corpus Version 2.0* is 89.56% by Sassano(2004)'s SR algorithm. [5]

Since we don't have the outputs of Sassano's experiments, we cannot do a McNemar test between the tournament model and Sassano's results. Our model outperforms Sassano's results by the dependency accuracy, but the difference between these two is not significant by prop test ($p = 0.097$).

When we add the additional and case particle features, the improvement of our model is less than that of other algorithms. This is interpreted that our model can consider richer contextual informa-

[5] This accuracy in Sassano (2004) is not for *Kyoto Text Corpus Version 4.0* but *Version 2.0* The feature set of Sassano's experiment is also different from our experiment.

Method	Features	Jan. 9th	Jan.10th	Jan. 15th
Tournament	Standard feature only	89.89/49.63	89.63/48.34	89.40/49.70
	All features	**90.09/49.71**	**90.11/49.02**	**90.35/52.59**
SR algorithm	Standard feature only	88.18/45.92	88.80/44.76	88.03/47.24
(Sassano, 2004)	All features	89.22/47.90	89.79/47.87	89.55/49.79
CC algorithm	Standard feature only	88.17/45.92	88.80/44.76	88.00/47.24
(Kudo and Matsumoto, 2002)	All features	89.22/47.90	89.80/47.94	89.53/49.79
CLE algorithm	Standard feature only	88.64/45.34	88.16/43.14	88.07/45.21
(McDonald et al., 2005)	Standard and Additional	89.21/46.83	89.05/45.03	88.90/48.43

Table 2: Dependency and sentence accuracy [%] using 7,587 sentences as training data.

tion within the algorithm itself than other models.

This result also shows that the accuracies of the SR algorithm and CC algorithm are comparable when using the same features. However, this does not mean that their substantial power is comparable because the parsing order limits the available dynamic features.

4.4 Parsing Speed

Parsing time and the size of the training examples are shown in Table 3. All features were used. The column "# Step" represents the number of SVM classification steps in parsing all the test data. Time complexity of the tournament model and CC algorithm are $O(n^2)$ and that of the SR algorithm is $O(n)$. The tournament model needs 1.7 times more SVM classification steps and is 4 times slower than the SR algorithm. The reason for this difference in steps (x1.7) and time (x4) is the number of training examples and features in the SVM classification.

4.5 Comparison to Relative Preference Method

We performed another experiment under the same settings as Kudo's (2005) to compare the tournament model and relative preference method. The corpus is *Kyoto Text Corpus Version 3.0* since Kudo and Matsumoto (2005) used this corpus. Training data is articles from January 1st to 11th and editorials from January to August (24,263 sentences). Test data is articles from January 14th to 17th and editorials from October to December (9,287 sentences). We did not perform parameter engineering by development data, although Kudo and Matsumoto (2005) performed it. The criteria for dependency accuracy are the same as the experiments above. However, the criteria for sentence accuracy in this section include all sentences, even if the length is one as Kudo and Matsumoto (2005) did.

The results are shown in Table 4. Note that Kudo and Matsumoto (2005) and our feature sets are different. Only the CC Algorithm is tested with both feature sets. Our feature set looks better than Kudo's. By McNemar test ($p < 0.01$) on the dependency accuracy, the tournament model outperforms both the SR and CC algorithms significantly. Since we don't have the outputs of relative preference methods, we cannot do a McNemar test between the tournament model and the relative preference methods. By prop test ($p < 0.01$) on the dependency accuracy, our model significantly outperforms the relative preference method of Kudo and Matsumoto (2005). Though our model outperforms the "combination" model of Kudo and Matsumoto (2005) by the dependency accuracy, the difference between these two is not significant by prop test ($p = 0.014$). [6]

Note that, a log-linear model is used in Kudo's experiment. The log-linear model has shorter training time than SVM. The log-linear model requires feature combination engineering by hand, while SVMs automatically consider the feature combination by the use of polynomial kernels.

5 Discussion and Future Work

In our error analysis, many errors are observed in coordination structures. Sassano (2004) reported that introduction of features of coordinated *bun-*

[6] The "combination" model is the combination of the CC algorithm and relative preference method. In Kudo's experiment, whereas the relative preference method outperforms the CC algorithm for long-distance relations, it is reversed for short-distance relations. They determined the optimal combination (the threshold set at *bunsetsu* length 3) using the development set. In our experiment, the tournament model outperforms the CC and SR algorithms for relations of all lengths. Therefore, the tournament model doesn't need such ad hoc combination.

Method	# Step	Time[s]	# Example	# Feature
Tournament	26396	371	374579	56165
SR algorithm (Sassano, 2004)	15641	80	94669	37183
CC algorithm (Kudo and Matsumoto, 2002)	18922	99	112759	37183

Table 3: Parsing time and the size of the training examples.

Method	Features	Dep. Acc.	Sentence Acc.
Tournament	All	**91.96**	**57.44**
SR algorithm (Sassano, 2004)	All	91.48	55.67
CC algorithm (Kudo and Matsumoto, 2002)	All	91.47	55.65
Combination – CC and Relative preference	Kudo's (2005)	91.66	56.30
Relative preference (Kudo and Matsumoto, 2005)	Kudo's (2005)	91.37	56.00
CC algorithm (Kudo and Matsumoto, 2002)	Kudo's (2005)	91.23	55.59

Table 4: Dependency and sentence accuracy [%] using 24,263 sentences as training data with all features: comparison with Kudo(2005)'s experiments.

setsu improves accuracy. In *Kyoto Text Corpus Version 4.0*, coordination and apposition are annotated with different types of dependency relation. We did not use this information in parsing. A simple extension is to include those dependency types. Another extension is to employ a coordination analyzer as a separate process as proposed by Shimbo and Hara (2007).

Incorporating co-occurrence information will also improve the parsing accuracy. One usage of such information is verb-noun co-occurrence information that would represent selectional preference for case-frame information. Abekawa and Okumura (2006) proposed a reranking method of *k*-best dependency analyzer outputs using co-occurrence information. We have already developed a method to output *k*-best dependency trees. One of our future works is to test the reranking method using co-occurrence information on the *k*-best dependency trees.

Multilingual parsing is another goal. Japanese is a strict head-final language. However, most languages do not have such constraint. A different parsing algorithm should be employed for other less constrained languages so as to relax this constraint. A simple solution is to introduce a discrimination model according to whether the head is on the left-hand-side or on the right-hand-side of a dependent. Existence of projective constraint does not matter for the tournament model. The tournament model can be extended to relax the projective constraint. The preliminary results for English are shown in our CoNLL Shared Task 2008 report (Watanabe et al., 2008). The unlabeled syntactic dependency accuracy of 90.73% for WSJ data shows that the model is also effective in other (not strictly head final, non-projective) languages. In parsing word sequences, $O(n^2)$ time complexity becomes a serious problem compared to parsing *bunsetsu* sequences. Since a *bunsetsu* is a base phrase in Japanese, the number of *bunsetsu's* is much less than the number of words. One solution is to perform base phrase chunking in advance and to apply dependency parsing on the base phrase sequences.

A reviewer pointed out similarities between our model and RankSVM. RankSVM compares pairs of elements to find out relative ordering between elements. Our tournament model is a special case where two elements are compared, but with a specific viewpoint of a focused dependent.

6 Conclusions

We proposed a Japanese dependency parsing algorithm using the tournament model. The tournament model is a 3-tuple *bunsetsu* model and improves discriminative performance of selecting correct head compared with the conventional 2-tuple models. The most likely candidate head is selected by one-on-one games in the step-ladder tournament. The proposed model considers the relative position between the nearer and farther candidates. The model also considers all candidate heads, which are not considered in deterministic parsing algorithms. The tournament model is robust for the free-order language. The accu-

racy of our model significantly outperforms that of the previous methods in most experiment settings. Even though the problem of parsing speed remains, our research showed that considering two or more candidate heads simultaneously can achieve more accurate parsing.

References

Abekawa, Takeshi and Manabu Okumura. 2006. Japanese Dependency Parsing Using Co-occurrence Information and a Combination of Case Elements. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL 2006)*, pages 833–840.

Buchholz, Sabine and Erwin Marsi. 2006. CoNLL-X Shared Task on Multilingual Dependency Parsing. In *CoNLL-2006: Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 149–164.

Chu, Yoeng-Jin and Tseng-Hong Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.

Crammer, Koby and Yoram Singer. 2003. Ultraconservative Online Algorithms for Multiclass Problems. *Journal of Machine Learning Research*, 3:951–991.

Edmonds, Jack. 1967. Optimum branchings. *Journal of Research of the Natural Bureau of Standards*, 71B:233–240.

Eisner, Jason M. 1996. Three New Probabilistic Models for Dependency Parsing: An Exploration. In *COLING-96: Proceedings of the 16th Conference on Computational Linguistics - Volume 1*, pages 340–345.

Iida, Ryu, Kentaro Inui, Hiroya Takamura, and Yuji Matsumoto. 2003. Incorporating Contextual Cues in Trainable Models for Coreference Resolution. In *EACL Workshop 'The Computational Treatment of Anaphora'*.

Kudo, Taku and Yuji Matsumoto. 2002. Japanese Dependency Analysis Using Cascaded Chunking. In *CoNLL-2002: Proceedings of the Sixth Conference on Computational Language Learning*, pages 1–7.

Kudo, Taku and Yuji Matsumoto. 2005. Japanese Dependency Parsing Using Relative Preference of Dependency (in Japanese). *Information Processing Society of Japan, Journal*, 46(4):1082–1092.

McDonald, Ryan, Koby Crammer, and Fernando Pereira. 2005. Online Large-Margin Training of Dependency Parsers. In *ACL-2005: Proceedings of 43rd Annual Meeting of the Association for Computational Linguistics*, pages 523–530.

Nivre, Joakim and Jens Nilsson. 2005. Psuedo-Projective Dependency Parsing. In *ACL-2005: Proceedings of 43rd Annual Meeting of the Association for Computational Linguistics*, pages 99–106.

Nivre, Joakim and Mario Scholz. 2004. Deterministic Dependency Parsing of English Text. In *COLING-2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 64–70.

Nivre, Joakim, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 Shared Task on Dependency Parsing. In *CoNLL-2007: Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL-2007*, pages 915–932.

Nivre, Joakim. 2003. An Efficient Algorithm for Projective Dependency Parsing. In *IWPT-2003: 8th International Workshop on Parsing Technology*, pages 149–160.

Sassano, Manabu. 2004. Linear-Time Dependency Analysis for Japanese. In *COLING-2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 8–14.

Shimbo, Masashi and Kazuo Hara. 2007. A Discriminative Learning Model for Coordinate Conjunctions. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 610–619.

Watanabe, Yotaro, Masakazu Iwatate, Masayuki Asahara, and Yuji Matsumoto. 2008. A Pipeline Approach for Syntactic and Semantic Dependency Parsing. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning (To Appear)*.

Yamada, Hiroyasu and Yuji Matsumoto. 2003. Statistical Dependency Analysis with Support Vector Machines. In *IWPT-2003: 8th International Workshop on Parsing Technology*, pages 195–206.