A Pipeline Approach for Syntactic and Semantic Dependency Parsing

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

Abstract

This paper describes our system for syntactic and semantic dependency parsing to participate the shared task of CoNLL-2008. We use a pipeline approach, in which syntactic dependency parsing, word sense disambiguation, and semantic role labeling are performed separately: Syntactic dependency parsing is performed by a tournament model with a support vector machine; word sense disambiguation is performed by a nearest neighbour method in a compressed feature space by probabilistic latent semantic indexing; and semantic role labeling is performed by a an online passive-aggressive algorithm. The submitted result was 79.10 macro-average F1 for the joint task, 87.18% syntactic dependencies LAS, and 70.84 semantic dependencies F1. After the deadline, we constructed the other configuration, which achieved 80.89 F1 for the joint task, and 74.53 semantic dependencies F1. The result shows that the configuration of pipeline is a crucial issue in the task.

1 Introduction

This paper presents the description of our system in CoNLL-2008 shared task. We split the shared task into five sub-problems – syntactic dependency parsing, syntactic dependency label classification, predicate identification, word sense disambiguation, and semantic role labeling. The overview of our system is illustrated in Figure 1. Our dependency parsing module is based on a tournament model (Iida et al., 2003), in which a dependency attachment is estimated in step-ladder tournament matches. The relative preference of the attachment is modeled by one-on-one match in the tournament. Iwatate et al. (Iwatate et al., 2008) initially proposed the method for Japanese dependency parsing, and we applied it to other languages by relaxing some constraints (Section 2.1). Dependency label classification is performed by a linear-chain sequential labeling on the dependency siblings like McDonald's schemata (McDonald et al., 2006). We use an online passive-aggressive algorithm (Crammer et al., 2006) for linear-chain sequential labeling (Section 2.2). We also use the other linear-chain sequential labeling method to annotate whether each word is a predicate or not (Section 2.3). If an identified predicate has more than one sense, a nearest neighbour classifier disambiguates the word sense candidates (Section 2.4). We use an online passive-aggressive algorithm again for the semantic role labeling (Section 2.5). The machine learning algorithms used in separated modules are diverse due to role sharing.[1]

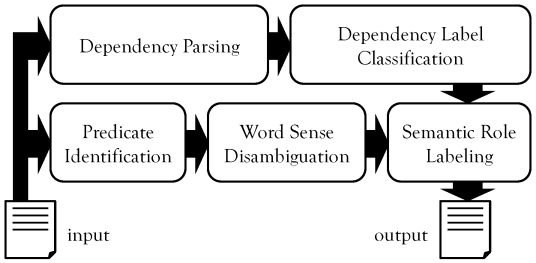

Figure 1: Overview of the System

[1] Unlabeled dependency parsing was done by Iwatate, dependency label classification and semantic role labeling was done by Watanabe, predicate identification and word sense

CoNLL 2008: Proceedings of the 12th Conference on Computational Natural Language Learning, pages 228–232
Manchester, August 2008

We attempt to construct a framework in which each module passes k-best solutions and the last semantic role labeling module performs reranking of the k-best solutions using the overall information. Unfortunately, we couldn't complete the framework before the deadline of the test run. Our method is not a "joint learning" approach but a pipeline approach.

2 Methods

2.1 Unlabeled Dependency Parsing

The detailed description of the tournament model-based Japanese dependency parsing is found in (Iwatate et al., 2008). The original Iwatate's parsing algorithm was for Japanese, which is for a strictly head-final language. We adapt the algorithm to English in this shared task. The tournament model chooses the most likely candidate head of each of the focused words in a step-ladder tournament. For a given word, the algorithm repeats to compare two candidate heads and finds the most plausible head in the series of a tournament. On each comparison, the winner is chosen by an SVM binary classifier with a quadratic polynomial kernel[2]. The model uses different algorithms for training example generation and parsing. Figures 2 and 3 show training example generation and parsing algorithm, respectively. Time complexity of both algorithms is $O(n^2)$ for the number of words in an input sentence. Below, we present the features for SVM

```
// N: # of tokens in input sentence
// true_head[j]: token j's head at
//                   training data
// gen(j,i1,i2,LEFT): generate an example
//   where token j is dependent of i1
// gen(j,i1,i2,RIGHT): generate an example
//   where token j is dependent of i2
// Token 0 is the virtual ROOT.

for j = 1 to N-1 do
  h = true_head[j];
  for i = 0 to h-1 do
      if i!=j then gen(j,i,h,RIGHT);

  for i = h+1 to N do
      if i!=j then gen(j,h,i,LEFT);
end-for;
```

Figure 2: Pseudo Code of Training Example Generation

```
// N: # of tokens in input sentence
// head[]: (analyzed-) head of tokens
// 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.
// cands.push_back(k): add token index k
//   to the end of cands.
// cands.erase(i): remove i-th element
//   from cands.

for j = 1 to N do
  cands = [];
  for i = 0 to N do
    if i!=j then cands.push_back(i);
  end-for;

  while cands.size() > 1 do
    if classify(j,cands[0],
            cands[1]) = LEFT then
      cands.erase(1);
    else
      cands.erase(0);
    end-if;
  end-while;

  head[j] = cands[0];
end-for;
```

Figure 3: Pseudo Code of Parsing Algorithm

in our tournament model. The FORM, LEMMA, GPOS(for training), PPOS(for testing, instead of GPOS), SPLIT_FORM, SPLIT_LEMMA, PPOSS in the following tokens were used as the features:

- Dependent, candidate1, candidate2
- Immediately-adjacent tokens of dependent, candidate1, candidate2, respectively
- All tokens between dependent-candidate1, dependent-candidate2, candidate1-candidate2, respectively

We also used the distance feature: distance (1 or 2-5 or 6+ tokens) between dependent-candidate1, dependent-candidate2, and candidate1-candidate2. Features corresponding to the candidates, including the distance feature, have a prefix that indicates its side: "L-"(the candidate appears on left-hand-side of the dependent) or "R-"(appears on right-hand-side of the dependent). Training an SVM model with all examples is time-consuming, and split the examples by the dependent GPOS for training (PPOS for testing, instead of GPOS[3]) to run SVM training in parallel. Since the number of examples with the dependent PPOS:IN, NN, NNP

disambiguation was done by Asahara, and all tasks were supervised by Matsumoto.

[2]We use TinySVM as an SVM classifier. chasen.org/~taku/software/TinySVM/

[3]We cannot use GPOS for testing due to the shared task regulation.

is still large, we used only first 1.5 million examples for the dependent GPOS. Note that, the algorithm does not check the well-formedness of dependency trees [4].

2.2 Dependency Label Classification

This phase labels a dependency relation label to each word in a parse tree produced in the preceding phase. (McDonald et al., 2006) suggests that edges of head x_i and its dependents $x_{j1}, ..., x_{jM}$ are highly correlated, and capturing these correlation improves classification accuracy. In their approach, edges of a head and its dependents $e_{i,j1}, ..., e_{i,jM}$ are classified sequentially, and then Viterbi algorithm is performed to find the highest scoring label sequence. We take a similar approach with some simplification. In our system, each edge is classified deterministically, and the previous decision is used as a feature for the subsequent classification.

We use an online passive aggressive algorithm (Crammer et al., 2006) [5] for dependency label classification since it converges fast, gives good performance and can be implemented easily. The features used in this phase are primarily similar to that of (McDonald et al., 2006).

Word features: SPLIT_LEMMA, PPOS, affix (lengths 2 and 3) of the head and the dependent.

Position: Position relation between the head and the dependent (Is the head anterior to dependent?). Is the word top of the sentence? Is the word last of the sentence?

Context features: SPLIT_LEMMA, PPOS, affix (lengths 2 and 3) of the nearest left/right word. SPLIT_LEMMA and PPOS bigram (ww, wp, pw, pp) of the head and the dependent (window size 5).

Sibling features: SPLIT_LEMMA, PPOS, affix (lengths 2 and 3) of the dependent's nearest left and right siblings in the dependency tree.

Other features: The number of dependent's children. Whether the dependent and the dependent's grand parent SPLIT_LEMMA/PPOS are the same. The previous classification result (previous label).

2.3 Predicate Identification

This phase solves which word can be a predicate. In the predicate spotting, the linear-chain CRF (Lafferty et al., 2001) annotates whether the word is a predicate or not. The FORM, LEMMA (itself, and whether the LEMMA is registered in the PropBank/NomBank frames), SPLIT_FORM, SPLIT_LEMMA, PPOSS within 5 token window size are used as the features. We also use bigram features within 3 token window size and trigram features within 5 token window size for FORM, LEMMA, SPLIT_FORM, SPLIT_LEMMA, and PPOSS. The main reason why we use a sequence labeling method for predicate identification was to relax the effect of the tagging error of PPOS and PPOSS. However, we will show later that this module aggravates the total performance.

2.4 Word Sense Disambiguation

For the word sense disambiguation, we use 1-nearest neighbour method in a compressed feature space by probabilistic latent semantic indexing (PLSI). We trained the word sense disambiguation model from the example sentences in the training/development data and PropBank/NomBank frames. The metric in the nearest neighbour method is based on the occurrence of LEMMA in the example sentences. However, the examples in the PropBank/NomBank do not contain the lemma information. To lemmatize the words in the PropBank/NomBank, we compose a lemmatizer from the FORM-LEMMA table in the training and development.[6] Since the metric space is very sparse, PLSI (Hofmann, 1999) is used to reduce the metric space dimensions. We used KL-divergence between two examples of $P(d_i|z_k)$ of $P(d_i, w_j) = \sum_k P(d_i|z_k)P(w_j|z_k)P(z_k)$ as hemi-metric for the nearest neighbour method[7], in which $d_i \in D$ is an example sentence in the training/devel/test data and PropBank/NomBank frames; $w_j \in W$ is LEMMA; and $z_k \in Z$ is a latent class. We use $|Z| = 100$, which gave the best performance in the development data. Note, we transductively used the test data for the PLSI modeling within the test run period.

2.5 Semantic Role Labeling

While semantic role labeling task is generally performed by two phases: argument identification and argument classification, we did not divide the task

[4] We tried to make a k-best cascaded model among the modules. The latter module can check the well-formedness of the tree. The current implementation skips this well-formedness checking.

[5] We use **PA** algorithm among **PA, PA-I and PA-II** in (Crammer et al., 2006).

[6] We are not violating the closed track regulation to build the lemmatizer. If a word in the PropBank/NomBank is not in the training/development data, we give up lemmatization.

[7] We use $D_{KL} = \sum_k P(d_{\texttt{input data}}|z_k) log \frac{P(d_{\texttt{input data}}|z_k)}{P(d_{\texttt{1-nearest data}}|z_k)}$ as hemi-metric. It is a non-commutative measure.

into the two phases. That is, argument candidates are directly assigned a particular semantic role label. We did not employ any candidate filtering procedure, so argument candidates consist of words in any predicate-word pair. The argument candidates that have no roles are assigned "NONE" label. For the reason that described in Section 2.2 (fast convergence and good performance), we use an online passive aggressive algorithm for learning the semantic role classifiers.

Useful features for argument classification of verb and noun predicates are different. For example, voice (active or passive) is essential for verb predicate's argument classification. On the other hand, presence of a genitive word is useful for noun predicate's argument classification. For this reason, we created two models: argument classifier for verb predicates and that for noun predicates.

Semantic frames are useful information for semantic role classification. Generally, obligatory arguments not included in semantic frames do not appear in actual texts. For this reason, we use PropBank/NomBank semantic frames for semantic role pruning. Suppose semantic roles in the semantic frame are $F_i = \{A0, A1, A2, A3\}$. Since obligatory arguments are $\{A0...AA\}$, the remaining arguments $\{A4, A5, AA\}$ are removed from label candidates.

For verb predicates, the features used in our system are based on (Hacioglu, 2004). We also employed some other features proposed in (Gildea and Jurafsky, 2002; Pradhan et al., 2004b). For noun predicates, the features are primarily based on (Pradhan et al., 2004a). The features that we defined for semantic role labeling are as follows:

Word features: SPLIT_LEMMA and PPOS of the predicate, dependent and dependent's head, and its conjunctions.

Dependency label: The dependency label between the argument candidate and the its head.

Family: The position of the argument candidate with respect to the predicate position over the dependency tree (e.g., child, sibling).

Position: The position of the head of the dependency relation with respect to the predicate position in the sentence.

Pattern: The left-to-right chain of the PPOS/dependency labels of the predicate's children.

Context features: PPOS of the nearest left/right word.

Path features: SPLIT_LEMMA, PPOS and dependency label paths between the predicate and the argument candidate, and its path bi-gram.

Distance: The number of paths between the predicate and the argument candidate.

Voice: Voice of the predicate (active or passive) and voice-position conjunction (for verb predicates).

Is predicate plural: Whether the predicate is singular or plural (for noun predicates).

Genitives between the predicate and the argument: Is there a genitive word between the predicate and the argument? (for noun predicates)

3 Results

Table 1 shows the result of our system. The proposed method was effective in dependency parsing (rank 3rd), but was not good in semantic role labeling (rank 9th). One reason of the result of semantic role labeling could be usages of PropBank/NomBank frames. We did not achieve the maximum use of the resources, hence the design of features and the choice of learning algorithm may not be optimal.

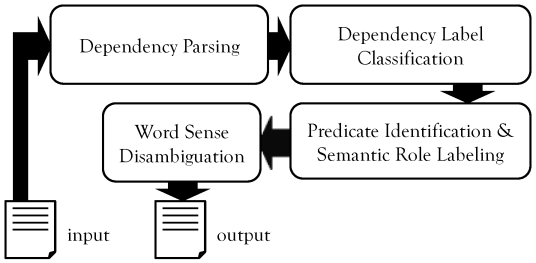

Figure 4: Overview of the Modified System

The other reason is the design of the pipeline. We changed the design of the pipeline after the test run. The overview of the modified system is illustrated in Figure 4. After the syntactic dependency parsing, we limited the predicate candidates as verbs and nouns by PPOSS, and filtered the argument candidates by Xue's method (Xue and Palmer, 2004). Next, the candidate pair of predicate-argument was classified by an online passive-aggressive algorithm as shown in Section 2.5. Finally, the word sense of the predicate is determined by the module in Section 2.4. The new result is scores with $*$ in Table 1. The result means that the first design was not the best for the task.

Acknowledgements

We would like to thank the CoNLL-2008 shared task organizers and the data providers (Surdeanu et al., 2008).

Problem	All	WSJ	Brown	Rank
Complete Problem	79.10 (80.89*)	80.30 (82.06*)	69.29 (71.32*)	9th
Semantic Dependency	70.84 (74.53*)	72.37 (76.01*)	58.21 (62.41*)	9th
Semantic Role Labeling	67.92 (72.31*)	69.31 (73.62*)	56.42 (61.64*)	-
Predicate Identification & Word Sense Disambiguation	77.20 (79.17*)	79.02 (80.99*)	62.10 (64.03*)	-
Syntactic Dependency (Labeled)	87.18	88.06	80.17	3rd
Syntactic Label Accuracy	91.63	92.31	86.26	-
Unlabeled Syntactic Dependency Unlabeled	90.20	90.73	85.94	-

The scores with $*$ mark are our post-evaluation results.

Table 1: The Results – Closed Challenge

References

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.

Crammer, Koby, Ofer Dekel, Joseph Keshet, Shai Shalev-Schwarz, and Yoram Singer. 2006. Online Passive-Agressive Algorithms. *Journal of Machine Learning Research*, 7:551–585.

Gildea, Daniel and Daniel Jurafsky. 2002. Automatic Labeling of Semantic Roles. *Computational Linguistics*, 28(3):245–288.

Hacioglu, Kadri. 2004. Semantic role labeling using dependency trees. In *COLING-2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 1273–1276.

Hofmann, Thomas. 1999. Probabilistic Latent Semantic Indexing. In *SIGIR-1999: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Informatino Retrieval*, pages 50–57.

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'*, pages 23–30.

Iwatate, Masakazu, Masayuki Asahara, and Yuji Matsumoto. 2008. Japanese Dependency Parsing Using a Tournament Model. In *COLING-2008: Proceedings of the 22nd International Conference on Computational Linguistics (To Appear)*.

Lafferty, John D., Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *ICML-1001: Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289.

McDonald, Ryan, Kevin Lerman, and Fernando Pereira. 2006. Multilingual Dependency Analysis with a Two-Stage Discriminative Parser. In *CoNLL-2006: Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 216–220.

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.

Pradhan, Sameer, Honglin Sun, Wayne Ward, James H. Martin, and Dan Jurafsky. 2004a. Parsing Arguments of Nominalizations in English and Chinese. In *HLT-NAACL-2004: Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 141–144.

Pradhan, Sameer, Wayne Ward, Kadri Hacioglu, James H. Martin, and Dan Jurafsky. 2004b. Shallow Semantic Parsing Using Support Vector Machines. In *HLT-NAACL-2004: Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 233–240.

Surdeanu, Mihai, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies. In *CoNLL-2008: Proceedings of the 12th Conference on Computational Natural Language Learning*.

Xue, Nianwen and Martha Palmer. 2004. Calibrating Features for Semantic Role Labeling. In *EMNLP-2004: Proceedings of 2004 Conference on Empirical Methods in Natural Language Processing*, pages 88–94.