Semantic Dependency Parsing of NomBank and PropBank: An Efficient Integrated Approach via a Large-scale Feature Selection [*]

Hai Zhao(赵海)[†], Wenliang Chen(陈文亮)[*], Chunyu Kit[†](揭春雨)
[†]Department of Chinese, Translation and Linguistics
City University of Hong Kong
Tat Chee Avenue, Kowloon, Hong Kong, China
[*]Language Infrastructure Group, MASTAR Project
National Institute of Information and Communications Technology
3-5 Hikari-dai, Seika-cho, Soraku-gun, Kyoto, Japan, 619-0289
haizhao@cityu.edu.hk, chenwl@nict.go.jp

Abstract

We present an integrated dependency-based semantic role labeling system for English from both NomBank and PropBank. By introducing assistant argument labels and considering much more feature templates, two optimal feature template sets are obtained through an effective feature selection procedure and help construct a high performance single SRL system. From the evaluations on the date set of CoNLL-2008 shared task, the performance of our system is quite close to the state of the art. As to our knowledge, this is the first integrated SRL system that achieves a competitive performance against previous pipeline systems.

1 Introduction

We investigate the possibility to construct an effective integrated system for dependency-based semantic role labeling (SRL) task. This means in this work that a single system handles all these sub-tasks, predicate identification/disambiguation and argument identification/classification, regardless of whether the predicate is verbal or nominal.

Traditionally, a SRL task, either dependency or constituent based, is implemented as two subtasks, namely, argument identification and classification. If the predicate is unknown, then a predicate identification or disambiguation subtask should be additionally considered. A pipeline framework is usually adopted to handle all these sub-tasks. The reason to divide the whole task into multiple stages is two-fold, one is each subtask asks for its favorable features, the other is at the consideration of computational efficiency. Generally speaking, a joint system is slower than a pipeline system in training. (Xue and Palmer, 2004) fount out that different features suited for different sub-tasks of SRL, i.e. argument identification and classification. The results from CoNLL shared tasks in 2005 and 2008 (Carreras and Marquez, 2005; Koomen et al., 2005; Surdeanu et al., 2008; Johansson and Nugues, 2008), further show that SRL pipeline may be one of the standard to achieve a state-of-the-art performance in practice.

In the recent years, most works on SRL, including two CoNLL shared task in 2004 and 2005, focus on verbal predicates with the availability of PropBank (Palmer et al., 2005). As a complement to PropBank, NomBank (Meyers et al., 2004) annotates nominal predicates and their corresponding semantic roles using similar semantic framework as PropBank. Though SRL for nominal predicates offers more challenge, it draws relatively little attention (Jiang and Ng, 2006).

(Pustejovsky et al., 2005) discussed the issue of merging various treebanks, including PropBank, NomBank, and others. The idea of merging these two different treebanks was implemented in the CoNLL-2008 shared task (Surdeanu et al., 2008). However, few empirical studies support the necessity of an integrated learning strategy from NomBank and PropBank. Though aiming at Chinese SRL, (Xue, 2006) reported that their experiments show that simply adding the verb data to the training set of NomBank and extracting the same features from the verb and noun instances will hurt the overall performance. From the results of CoNLL-2008 shared task, the top system by (Johansson and Nugues, 2008) also used two

[*]This study is partially supported by CERG grant 9040861 (CityU 1318/03H), CityU Strategic Research Grant 7002037.

different subsystems to handle verbal and nominal predicates, respectively.

Despite all the above facts, an integrated SRL system still holds some sort of merits, being easier to implement, a single-stage feature selection benefiting the whole system, an all-in-one model outputting all required semantic role information and so on.

The shared tasks at the CoNLL 2008 and 2009 are devoted to the joint learning of syntactic and semantic dependencies, which show that SRL can be well performed using only dependency syntax input. Using data and evaluation settings of the CoNLL-2008 shared task, this work will only focus on semantic dependency parsing and compares the best-performing SRL system in the CoNLL-2009 shared Task (Zhao et al., 2009b) with those in the CoNLL-2008 shared task (Surdeanu et al., 2008; Hajič et al., 2009)[1].

Aiming at main drawbacks of an integrated approach, two key techniques will be applied. 1) Assistant argument labels are introduced for the further improvement of argument pruning. This helps the development of a fast and lightweight SRL system. 2) Using a greedy feature selection algorithm, a large-scale feature engineering is performed on a much larger feature template set than that in previous work. This helps us find features that may be of benefit to all SRL sub-tasks as long as possible. As two optimal feature template sets have been proven available, for the first time we report that an integrated SRL system may provide a result close to the state-of-the-art achieved by those SRL pipelines or individual systems for some specific predicates.

2 Adaptive Argument Pruning

A word-pair classification is used to formulate semantic dependency parsing as in (Zhao and Kit, 2008). As for predicate identification or disambiguation, the first word is set as a virtual root (which is virtually set before the beginning of the sentence.) and the second as a predicate candidate. As for argument identification/classification, the first word in a word pair is specified as a predicate candidate and the second as an argument candidate. In either of case, the first word is called a semantic head, and noted as p in our feature representation, the second is called a semantic dependent and noted as a.

Word pairs are collected for the classifier in such order. The first word of the pair is set to the virtual root at first, the second word is then specified as a predicate candidate. According to the result that the predicate candidate is classified or proven to be non-predicate, 1) the second word is reset to next predicate candidate if the answer is non-predicate, otherwise, 2) the first word of the pair is reset to the predicate that is just determined, and the second is set to every argument candidates one by one. The classifier will scan the input sentence from left to right to check if each word is a true predicate.

Without any constraint, all word pairs in an input sequence must be considered by the classifier, leading to poor computational efficiency and unnecessary performance loss. Thus, the training sample for SRL task needs to be pruned properly.

We use a simple strategy to prune predicate candidates, namely, only verbs and nouns are chosen in this case.

There are two paths to collect argument candidates over the sequence. One is based on an input syntactic dependency tree, the other is based on a linear path of the sentence. As for the former (hereafter it is referred to *synPth*), we continue to use a dependency version of the pruning algorithm of (Xue and Palmer, 2004). The pruning algorithm is readdressed as the following.

Initialization: Set the given predicate as the current node;

(1) The current node and all of its syntactic children are selected as argument candidates (children are traversed from left to right.).

(2) Reset the current node to its syntactic head and repeat step (1) until the root is reached.

Note that this pruning algorithm is slightly different from that of (Xue and Palmer, 2004), the predicate itself is also included in the argument candidate list as the nominal predicate sometimes takes itself as its argument.

The above pruning algorithm has been shown effective. However, it is still inefficient for a SRL

[1]CoNLL-2008 is an English-only task, while CoNLL-2009 is a multilingual one. Though the English corpus in CoNLL-2009 is almost identical to the corpus in the CoNLL-2008 shared task evaluation, the latter holds more sophisticated input structure as in (Surdeanu et al., 2008). The most difference for these two tasks is that the identification of semantic predicates is required in the task of CoNLL-2008 but not in CoNLL-2009.

system that needs to tackle argument identification/classification in a single stage. Assuming that arguments trend to surround their predicate, an assistant argument label '*_NoMoreArgument*' is introduced for further pruning. If an argument candidate in the above algorithm is assigned to such a label, then the pruning algorithm will end immediately. In training, this assistant label means no more samples will be generated for the current predicate, while in test, the decoder will not search arguments any more. It will be seen that this adaptive technique more effectively prunes argument candidates without missing more true arguments.

Along the linear path (hereafter referred to *linPth*), the classifier will search all words before and after the predicate. Similar to the pruning algorithm for *synPth*, we also introduce two assistant argument labels '*_noLeft*' and '*_noRight*' to adaptively prune words too far away from the predicate.

To show how assistant argument labels actually work, we give an example for $linPth$. Suppose an input sequence with argument labels for a predicate is

a	b	c	d	e	f	g	h	.
_	_	_	A1	_	A0	_	_	_

Note that c and g are two boundary words as no more arguments appear before or after them. After two assistant argument labels are added, it will be

a	b	c	d	e	f	g	h	.
_	_	*_noLeft*	A1	_	A0	*_noRight*	_	_

Training samples will generated from c to g according to the above sequence.

We use a Maximum Entropy classifier with a tunable Gaussian prior as usual. Our implementation of the model adopts L-BFGS algorithm for parameter optimization.

3 Feature Templates

3.1 Elements for Feature Generation

Motivated by previous works, we carefully consider those factors from a wide range of features that can help semantic role labeling for both predicate disambiguation, argument's identification and classification as the predicate is either verbal or nominal. These works include (Gildea and Jurafsky, 2002; Carreras and Marquez, 2005; Koomen et al., 2005; Marquez et al., 2005; Dang and Palmer, 2005; Pradhan et al., 2005; Toutanova et al., 2005; Jiang and Ng, 2006; Liu and Ng, 2007; Surdeanu et al., 2007; Johansson and Nugues, 2008; Che et al., 2008). Most feature templates that we will adopt for this work will come from various combinations or integrations of the following basic elements.

Word Property. This type of elements include word form (*form* and its split form, *spForm*)[2], lemma (*lemma*,*spLemma*), and part-of-speech tag (*pos*, *spPos*), syntactic dependency label (*dprel*), and semantic dependency label (*semdprel*)[3].

Syntactic Connection. This includes syntactic head (h), left(right) farthest(nearest) child (lm, ln, rm, and rn), and high(low) support verb or noun. We explain the last item, support verb(noun). From a given word to the syntactic root along the syntactic tree, the first verb/noun/preposition that is met is called as its low support verb/noun/preposition, and the nearest one to the root is called as its high support verb/noun/preposition. The concept of support verb was broadly used (Toutanova et al., 2005; Xue, 2006; Jiang and Ng, 2006)[4], we here extend it to nouns and prepositions. In addition, we introduce a slightly modified syntactic head, $pphead$, it returns the left most sibling of a given word if the word is headed by a preposition, otherwise it returns the original head.

Path. There are two basic types of path between the predicate and the argument candidates. One is the linear path (*linePath*) in the sequence, the other is the path in the syntactic parsing tree (*dpPath*). For the latter, we further divide it into four sub-types with respect to the syntactic root, *dpPath* is the full path in the syntactic tree. Leading two paths to the root from the predicate and the argument, respectively, the common part of these two paths will be *dpPathShare*. Assume that *dpPathShare* starts from a node r', then *dpPathPred* is from the predicate to r', and *dpPathArgu* is from the argument to r'.

Family. Two types of children sets for the predicate or argument candidate are considered, the

[2]In CoNLL-2008, Treebank tokens are split at the position that a hyphen (-) or a forward slash (/) occurs. This leads to two types of feature columns, non-split and split.

[3]Lemma and pos for either training or test are from automatically pre-analyzed columns in the input files.

[4]Note that the meaning of support verb is slightly different between (Toutanova et al., 2005) and (Xue, 2006; Jiang and Ng, 2006)

first includes all syntactic children (*children*), the second also includes all but excludes the left most and the right most children (*noFarChildren*).

Concatenation of Elements. For all collected elements according to *linePath*, *children* and so on, we use three strategies to concatenate all those strings to produce the feature value. The first is *seq*, which concatenates all collected strings without doing anything. The second is *bag*, which removes all duplicated strings and sort the rest. The third is *noDup*, which removes all duplicated neighbored strings.

We address some other elements that are not included by the above description as the following.

dpTreeRelation. It returns the relationship of a and p in the input syntactic tree. The possible values for this feature include `parent`, `sibling` etc.

isCurPred. It judges if a given word is the current predicate. If the word is the predicate, then it returns the predicate itself, otherwise it returns a default value.

existCross. It judges if a forthcoming dependency relation that is between a given word pair may cause any cross with all existing dependency relations.

distance. It counts the number of words along a given path, either *dpPath* or $linePath$.

existSemdprel. It checks if the given argument label for other predicates has been assigned to a given word.

voice. This feature returns *Active* or *Passive* for verbs, and a default value for nouns.

baseline. Two types of semantic role baseline outputs are used for features from (Carreras and Marquez, 2005)[5]. $baseline\_Ax$ tags the head of the first NP before the predicate as $A0$ and the head of the first NP after the predicate as $A1$. $baseline\_Mod$ tags the dependant of the predicate as *AM-MOD* as it is a modal verb.

We show some feature template examples derived from the above mentioned items.

a.lm.lemma The lemma of the left most child of the argument candidate.

p.h.dprel The dependant label of the syntactic head of the predicate candidate.

p_{-1}*.pos+p.pos pos* of the previous word of the predicate and *PoS* of the predicate itself.

a:p$|$*dpPath.lemma.bag* Collect all lemmas along the syntactic tree path from the argument to the predicate, then removed all duplicated ones and sort the rest, finally concatenate all as a feature string.

a:p.highSupportNoun$|$*linePath.dprel.seq* Collect all dependant labels along with the line path from the argument to the high support noun of the predicate, then concatenate all as a feature string.

3.2 Feature Template Selection

Based on the above mentioned elements, 781 feature templates (hereafter the set of these templates is referred to FT)[6] are initially considered. Feature templates in this initial set are constructed in a generalized way. For example, if we find that a feature template *a.lm.lemma* was once used in some existing work, then such three templates, *a.rm.lemma*, *a.rn.lemma*, *a.ln.lemma* will be also added into the set.

As an optimal feature template subset cannot be expected to be extracted from so large a set by hand, a greedy feature selection similar to that in (Jiang and Ng, 2006; Ding and Chang, 2008) is applied. The detailed algorithm is described in Algorithm 1. Assuming that the number of feature templates in a given set is n, the algorithm of (Ding and Chang, 2008) requires $O(n^2)$ times of training/test routines, it cannot handle a set that consists of hundreds of templates. As the time complexity of Algorithm 1 is only $O(n)$, it permits a large scale feature selection accomplished by paying a reasonable time cost. Though the time complexity of the algorithm given by (Jiang and Ng, 2006) is also linear, it should assume all feature templates in the initial selected set 'good' enough and handles other feature template candidates in a strict incremental way. However, these two constraints are not easily satisfied in our case, while Algorithm 1 may release these two constraints.

Choosing the first 1/10 templates in FT as the initial selected set S, the feature selection is performed for two argument candidate traverse schemes, $synPth$ and $linPth$, respectively. 4686 machine learning routines run for the former, while 6248 routines for the latter. Two feature template sets, FT_{syn} and FT_{lin}, are obtained at last. These two sets are given in Table 1-3. We see that two sets share 30 identical feature templates as in Table 1. FT_{syn} holds 51 different templates

[5]These baseline rules were developed by Erik Tjong Kim Sang, from the University of Antwerp, Belgium.

[6]This set with detailed explanation will be available at our website.

-	*p.lm.dprel*
-	*p.rm.dprel*
-	*p.spForm*
-	p_{-1}*.spLemma*
-	*p.spLemma*
-	p_{-1}*.spLemma+p.spLemma*
-	*p.spLemma +* p_1*.spLemma*
-	*p.spLemma + p.h.spForm*
-	*p.spLemma + p.currentSense*
-	*p.lemma*
-	*p.lemma +* p_1*.lemma*
-	p_{-1}*.pos+p.pos*
-	*a.isCurPred.lemma*
-	a_{-2}*.isCurPred.lemma +* a_{-1}*.isCurPred.lemma*
-	*a.isCurPred.spLemma*
-	a_{-1}*.isCurPred.spLemma + a.isCurPred.spLemma*
-	*a.isCurPred.spLemma +* a_1*.isCurPred.spLemma*
-	*a.children.dprel.bag*
-	a_{-1}*.spLemma + a.spLemma*
-	a_{-1}*.spLemma + a.dprel*
-	a_{-1}*.spLemma + a.dprel + a.h.spLemma*
-	*a.lm*$_{-1}$*.spLemma*
-	*a.rm*$_{-1}$*.dprel + a.spPos*
-	a_{-1}*.lemma + a.dprel + a.h.lemma*
-	*a.lemma + p.lemma*
-	*a.pos + p.pos*
-	*a.spLemma + p.spLemma*
-	*a:p|dpPath.dprel*
-	*a:p|dpPathArgu.dprel*
-	*a:p|dpPathPred.spPos*

Table 1: Feature templates for both $synPth$ and $linPth$

as in Table 2 and FT_{lin} holds 57 different templates as in Table 3. In these tables, the subscripts -2(or -1) and 1(or 2) stand for the previous and next words, respectively. For example, *a.lm*$_{-1}$*.lemma* returns the lemma of the previous word of the argument's left most child.

4 Decoding

After the predicate sense is disambiguated, an optimal argument structure for each predicate is determined by the following maximal probability.

$$S_p = \text{argmax} \prod_i P(a_i|a_{i-1}, a_{i-2}, ...), \quad (1)$$

where S_p is the argument structure, $P(a_i|a_{i-1}...)$ is the conditional probability to determine the label of the i-th argument candidate label. A beam search algorithm is used to find the optimal argument structure.

5 Evaluation Results

Our evaluation is performed on the standard training/development/test corpus of CoNLL-2008 shared task. The data is derived by merging a dependency version of the Penn Treebank with PropBank and NomBank. More details on the data are

Algorithm 1 Greedy Feature Selection

```
Input:
The set of all feature templates: $FT$
The set of selected feature templates: $S_0$

Output:
The set of selected feature templates: $S$

Procedure:
  Let the counter $i = 1$
  Let $S_i = S_0$ and $C = FT - S_i$
  while do
    Train a model with features according to $S_i$, test on development set and the result is $p_i$.
    Let $C_r = null$.
    for each feature template $f_j$ in set $S_i$ do
      Let $S' = S_i - f_j$.
      Train a model with features according to $S'$, test on development set and the result is $p'$.
      if $p' > p_i$ then
        $C_r = C_r + f_j$.
      end if
    end for
    $C = C + C_r$
    $S_i = S_i - C_r$
    Let $S'_i = S_i$
    Train a model with features according to $S'_i$, test on development set and the result is $q_i$.
    Let $C_r = null$
    for each feature template $f_j$ in set $C$ do
      Let $C' = S'_i + f_j$.
      Train a model with features according to $C'$, test on development set and the result is $p'$.
      if $p' > q_i$ then
        $C_r = C_r + f_j$.
      end if
    end for
    $C = C - C_r$
    $S'_i = S'_i + C_r$
    if $S_i = S_{i-1}$(No feature templates are added or removed) or, neither $p_i$ nor $q_i$ is larger than $p_{i-1}$ and $q_{i-1}$ then
      Output $S = \text{argmax}_{p_i,q_i}\{S_i, S'_i\}$ and the algorithm ends.
    else
      Let $i = i + 1$, $S_i$=$S_{i-1}$ and $C = FT - S_i$
    end if
  end while
```

–	*p_{-1}.lemma + p.lemma*
–	*p_{-2}.pos*
–	*p.pos*
–	*p_{-2}.spForm + p_{-1}.spForm*
–	*p_{1}.spForm*
–	*p.spForm + p.children.dprel.noDup*
–	*p.lm.spPos*
–	*p.spForm + p.lm.spPos + p.noFarChildren.spPos.bag + p.rm.spPos*
–	*p.dprel*
–	*p.children.dprel.bag*
–	*p.children.pos.seq*
–	*p.dprel = OBJ ?* [a]
–	*a.dprel*
–	*a_{-1}.lemma + a_{1}.lemma*
–	*a_{1}.lemma*
–	*a_{-1}.pos*
–	*a_{1}.spPos*
–	*a.h.lemma*
–	*a.h.spLemma*
–	*a.pphead.lemma*
–	*a.pphead.spLemma*
–	*a.lm.dprel + a.spPos*
–	*$a.rm_{-1}$.pos*
–	*a.spLemma + a.h.spPos*
–	*a.existSemdprel_A1*
–	*a.dprel = OBJ ?*
–	*a.form + a.children.pos.seq*
–	*a.children.adv.bag*[b]
–	*a:p|linePath.distance*
–	*a:p|dpPath.distance*
–	*a:p|existCross*
–	*a:p|dpPath.dprel.bag*
–	*a:p|dpPathPred.dprel.bag*
–	*a:p|dpPath.spForm.seq*
–	*a:p|dpPathArgu.spForm.seq*
–	*a:p|dpPathPred.spForm.bag*
–	*a:p|dpPath.spLemma.seq*
–	*a:p|dpPathArgu.spLemma.seq*
–	*a:p|dpPathArgu.spLemma.bag*
–	*a:p|dpPathPred.spLemma.bag*
–	*a:p|dpPath.spPos.bag*
–	*a:p|dpPathPred.spPos.bag*
–	*(a:p|dpPath.dprel.seq) + p.spPos*
–	*(a:p|dpTreeRelation) + a.spPos*
–	*(a:p|dpTreeRelation) + p.spPos*
–	*(a.highSupportVerb:p|dpTreeRelation) + a.spPos*
–	*a.highSupportNoun:p|dpPath.dprel.seq*
–	*a.lowSupportVerb:p|dpPath.dprel.seq*
–	*a:p|linePath.spForm.bag*
–	*a:p|linePath.spLemma.bag*
–	*a:p|linePath.spLemma.seq*

[a]This feature checks if the dependant type is *OBJ*.
[b]*adv* means all adverbs.

Table 2: Feature templates only for *synPth*

–	*p.currentSense + a.spLemma*
–	*p.currentSense + a.spPos*
–	*p.voice + (a:p|direction)*
–	*p.rm.dprel*
–	*p.children.dprel.noDup*
–	*p.rm.form*
–	*p.lowSupportNoun.spForm*
–	*p.lowSupportProp:p|dpTreeRelation*
–	*p_{-2}.form + p_{-1}.form*
–	*p.voice*
–	*p.form + p.children.dprel.noDup*
–	*p.pos + p.dprel*
–	*p.spForm + p.children.dprel.bag*
–	*a.voice + (a:p|direction)*
–	*a_{-1}.isCurPred.lemma*
–	*a_{1}.isCurPred.lemma*
–	*a_{-1}.isCurPred.lemma + a.isCurPred.lemma*
–	*a.isCurPred.lemma + a_{1}.isCurPred.lemma*
–	*a_{1}.isCurPred.spLemma*
–	*a_{-2}.isCurPred.spLemma + a_{-1}.isCurPred.spLemma*
–	*a.baseline_Ax + a.voice + (a:p|direction)*
–	*a.baseline_Mod*
–	*a.h.children.dprel.bag*
–	*a.lm.dprel + a.dprel*
–	*a.lm.dprel + a.pos*
–	*$a.lm_{-1}$.lemma*
–	*a.lm.lemma*
–	*$a.lm_{1}$.lemma*
–	*a.lm.pos + a.pos*
–	*a.lm.spForm*
–	*$a.lm_{-1}$.spPos*
–	*a.lm.spPos*
–	*a.ln.dprel + a.pos*
–	*a.noFarChildren.spPos.bag + a.rm.spPos*
–	*a.children.spPos.seq + p.children.spPos.seq*
–	*a.rm.dprel + a.pos*
–	*$a.rm_{-1}$.spPos*
–	*a.rm.spPos*
–	*$a.rm_{1}$.spPos*
–	*a.rn.dprel + a.spPos*
–	*a.form*
–	*a.form + a_{1}.form*
–	*a.form + a.pos*
–	*a_{-1}.lemma*
–	*a_{-1}.lemma + a.lemma*
–	*a_{-2}.pos*
–	*a.spForm + a_{1}.spForm*
–	*a.spForm + a.spPos*
–	*a.spLemma + a_{1}.spLemma*
–	*a.spForm + a.children.spPos.seq*
–	*a.spForm + a.children.spPos.bag*
–	*a.spLemma + a.h.spForm*
–	*a.spLemma + a.pphead.spForm*
–	*a.existSemdprel_A2*
–	*a:p|dpPathArgu.pos.seq*
–	*a:p|dpPathPred.dprel.seq*
–	*a:p|dpTreeRelation*

Table 3: Feature templates only for *linPth*

in (Surdeanu et al., 2008). Note that CoNLL-2008 shared task is essentially a joint learning task for both syntactic and semantic dependencies, however, we will focus on semantic part of this task. The main semantic measure that we adopt is semantic labeled F_1 score (Sem-F_1). In addition, the macro labeled F_1 scores (Macro-F_1), which was used for the ranking of the participating systems of CoNLL-2008, the ratio between labeled F_1 score for semantic dependencies and the LAS for syntactic dependencies (Sem-F_1/LAS), are also given for reference.

5.1 Syntactic Dependency Parsers

We consider three types of syntactic information to feed the SRL task. One is gold-standard syntactic input, and other two are based on automatically parsing results of two parsers, the state-of-the-art syntactic parser described in (Johansson and Nugues, 2008)[7](it is referred to *Johansson*) and an integrated parser described as the following (referred to MST_{ME}).

The parser is basically based on the MSTParser[8] using all the features presented by (McDonald et al., 2006) with projective parsing. Moreover, we exploit three types of additional features to improve the parser. 1) Chen et al. (2008) used features derived from short dependency pairs based on large-scale auto-parsed data to enhance dependency parsing. Here, the same features are used, though all dependency pairs rather than short dependency pairs are extracted along with the dependency direction from training data rather than auto-parsed data. 2) Koo et al. (2008) presented new features based on word clusters obtained from large-scale unlabeled data and achieved large improvement for English and Czech. Here, the same features are also used as word clusters are generated only from the training data. 3) Nivre and McDonald (2008) presented an integrating method to provide additional information for graph-based and transition-based parsers. Here, we represent features based on dependency relations predicted by transition-based parsers for the MSTParer. For the sake of efficiency, we use a fast transition-based parser based on maximum entropy as in Zhao and Kit (2008). We still use the similar feature notations of that work.

[7]It is a 2-order maximum spanning tree parser with pseudo-projective techniques. A syntactic-semantic reranking was performed to output the final results according to (Johansson and Nugues, 2008). However, only 1-best outputs of the parser before reranking are used for our evaluation. Note that the reranking may slightly improve the syntactic performance according to (Johansson and Nugues, 2008).

[8]It's freely available at http://mstparser.sourceforge.net.

Parser	Path	*Adaptive Pruning*		Coverage
		/wo	/w	Rate
Gold	$synPth$	2.13M	1.05M (49.30%)	98.4%
	$linPth$	5.29M	1.57M (29.68%)	100.0%
Johansson	$synPth$	2.15M	1.06M (49.30%)	95.4%
	$linPth$	5.28M	1.57M (29.73%)	100.0%
MST_{ME}	$synPth$	2.15M	1.06M (49.30%)	95.0%
	$linPth$	5.29M	1.57M (29.68%)	100.0%

Table 4: The number of training samples on argument candidates

Syn-Parser	LAS	*synPth*+FT_{syn}		*linPth*+FT_{lin}	
		Sem F_1	Sem-F_1 /LAS	Sem F_1	Sem-F_1 /LAS
MST_{ME}	88.39	80.53	91.10	79.83	90.31
Johansson	89.28	80.94	90.66	79.84	89.43
Gold	100.00	84.57	84.57	83.34	83.34

Table 5: Semantic Labeled F_1

5.2 The Results

At first, we report the effectiveness of the proposed adaptive argument pruning. The numbers of argument candidates are in Table 4. The statistics is conducted on three different syntactic inputs. The coverage rate in the table means the ratio of how many true arguments are covered by the selected pruning scheme. Note that the adaptive pruning of argument candidates using assistant labels does not change this rate. This ratio only depends on which path, either $synPth$ or $linPth$, is chosen, and how good the syntactic input is (if $synPth$ is the case). From the results, we see that more than a half of argument candidates can be effectively pruned for $synPth$ and even 2/3 for $linPth$. As mentioned by (Pradhan et al., 2004), argument identification plays a bottleneck role in improving the performance of a SRL system. The effectiveness of the proposed additional pruning techniques may be seen as a significant improvement over the original algorithm of (Xue and Palmer, 2004). The results also indicate that such an assumption holds that arguments trend to close with their predicate, at either type of distance, syntactic or linear.

Based on different syntactic inputs, we obtain different results on semantic dependency parsing as shown in Table 5. These results on different syntactic inputs also give us a chance to observe how semantic performance varies according to syntactic performance. The fact from the results is that the ratio Sem-F_1/LAS becomes relatively smaller as the syntactic input becomes better. Though not so surprised, the results do show that the argument traverse scheme $synPth$ always outperforms the other $linPth$. The result of this comparison partially shows that an integrated semantic role labeler is sensitive to the order of how argument candidates are traversed to some extent.

The performance given by $synPth$ is compared to some other systems that participated in the CoNLL-2008 shared task. They were chosen among the 20 participating systems either because they held better results (the first four participants) or because they used some joint learning techniques (Henderson et al., 2008). The results of (Titov et al., 2009) that use the similar joint learning technique as (Henderson et al., 2008) are also included[9]. Results of these evaluations on the test set are in Table 6. Top three systems of CoNLL-2008, (Johansson and Nugues, 2008; Ciaramita et al., 2008; Che et al., 2008), used SRL pipelines.

In this work, we partially use the similar techniques ($synPth$) for our participation in the shared tasks of CoNLL-2008 and 2009 (Zhao and Kit, 2008; Zhao et al., 2009b; Zhao et al., 2009a). Here we report that all SRL sub-tasks are tackled in one integrated model, while the predicate disambiguation sub-task was performed individually in both of our previous systems. Therefore, this is our first attempt at a full integrated SRL system.

(Titov et al., 2009) reported the best result by using joint learning technique up to now. The comparison indicates that our integrated system outputs a result quite close to the state-of-the-art by the pipeline system of (Johansson and Nugues, 2008) as the same syntactic structure input is adopted. It is worth noting that our system actually competes with two independent sub-systems of (Johansson and Nugues, 2008), one for verbal predicates, the other for nominal predicates. In addition, the results of our system is obtained without using additional joint learning technique like syntactic-semantic reranking. It indicates that our system is expected to obtain some further performance improvement by using such techniques.

[9]In addition, the work of (Henderson et al., 2008) and (Titov et al., 2009) jointly considered syntactic and semantic dependencies, that is significantly different from the others.

6 Conclusion

We have described a dependency-based semantic role labeling system for English from NomBank and PropBank. From the evaluations, the result of our system is quite close to the state of the art. As to our knowledge, it is the first integrated SRL system that achieves such a competitive performance against previous pipeline systems.

According to the path that the word-pair classifier traverses argument candidates, two integration schemes are presented. Argument candidate pruning and feature selection are performed on them, respectively. These two schemes are more than providing a trivial comparison. As assistant labeled are introduced to help further argument candidate pruning, and this techniques work well for both schemes, it support the assumption that arguments trend to surround their predicate. The proposed feature selection procedure also work for both schemes and output quite different two feature template sets, and either of the sets helps the system obtain a competitive performance, this fact suggests that the feature selection procedure is robust and effective, too.

Either of the presented integrated systems can provide a competitive performance. This conclusion about basic learning scheme for SRL is some different from previous literatures. However, according to our results, there does exist a ‘harmony’ feature template set that is helpful to both predicate and argument identification/classification, or SRL for both verbal and nominal predicates. We attribute this different conclusion to two main factors, 1) much more feature templates (for example, ten times more than those used by Xue et al.) than previous that are considered for a successful feature engineering, 2) a maximum entropy classifier makes it possible to accept so many various features in one model. Note that maximum entropy is not so sensitive to those (partially) overlapped features, while SVM and other margin-based learners are not so.

Acknowledgements

Our thanks give to Dr. Richard Johansson, who kindly provided the syntactic output for his participation in the CoNLL-2008 shared task.

Systems[a]	LAS	Sem-F_1	Macro F_1	Sem-F_1 /LAS	pred-F_1[b]	argu-F_1[c]	Verb-F_1[d]	Nomi-F_1[e]
Johansson:2008*[f]	**89.32**	**81.65**	**85.49**	**91.41**	**87.22**	**79.04**	**84.78**	77.12
Ours:*Johansson*	89.28	80.94	85.12	90.66	86.57	78.30	83.66	76.93
Ours:MST_{ME}	88.39	80.53	84.93	91.10	86.80	77.60	82.77	**77.23**
Johansson:2008	**89.32**	80.37	84.86	89.98	85.40	78.02	84.45	74.32
Ciaramita:2008*	87.37	78.00	82.69	89.28	83.46	75.35	80.93	73.80
Che:2008	86.75	78.52	82.66	90.51	85.31	75.27	80.46	75.18
Zhao:2008*	87.68	76.75	82.24	87.53	78.52	75.93	78.81	73.59
Ciaramita:2008	86.60	77.50	82.06	89.49	83.46	74.56	80.15	73.17
Titov:2009	87.50	76.10	81.80	86.97	–	–	–	–
Zhao:2008	86.66	76.16	81.44	87.88	78.26	75.18	77.67	73.28
Henderson:2008*	87.64	73.09	80.48	83.40	81.42	69.10	75.84	68.90
Henderson:2008	86.91	70.97	79.11	81.66	79.60	66.83	73.80	66.26
Ours:*Gold*	100.0	84.57	92.20	84.57	87.67	83.15	88.71	78.39

[a]Ranking according to Sem-F_1
[b]Labeled F_1 for predicate identification and classification
[c]Labeled F_1 for argument identification and classification
[d]Labeled F_1 for verbal predicates
[e]Labeled F_1 for nominal predicates
[f]* means post-evaluation results, which are available at the official website of CoNLL-2008 shared task, http://www.yr-bcn.es/dokuwiki/doku.php?id=conll2008:start.

Table 6: Comparison of the best existing systems

References

Xavier Carreras and Lluis Marquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of CoNLL-2005*, pages 152–164, Ann Arbor, Michigan, USA.

Wanxiang Che, Zhenghua Li, Yuxuan Hu, Yongqiang Li, Bing Qin, Ting Liu, and Sheng Li. 2008. A cascaded syntactic and semantic dependency parsing system. In *Proceedings of CoNLL-2008*, pages 238–242, Manchester, England, August.

Wenliang Chen, Daisuke Kawahara, Kiyotaka Uchimoto, Yujie Zhang, and Hitoshi Isahara. 2008. Dependency parsing with short dependency relations in unlabeled data. In *Proceedings of IJCNLP-2008*, Hyderabad, India, January 8-10.

Massimiliano Ciaramita, Giuseppe Attardi, Felice Dell'Orletta, and Mihai Surdeanu. 2008. Desrl: A linear-time semantic role labeling system. In *Proceedings of CoNLL-2008*, pages 258–262, Manchester, England, August.

Hoa Trang Dang and Martha Palmer. 2005. The role of semantic roles in disambiguating verb senses. In *Proceedings of ACL-2005*, pages 42–49, Ann Arbor, USA.

Weiwei Ding and Baobao Chang. 2008. Improving chinese semantic role classification with hierarchical feature selection strategy. In *Proceedings of EMNLP-2008*, pages 324–323, Honolulu, USA.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the 13th Conference on Computational Natural Language Learning (CoNLL-2009), June 4-5*, pages 1–18, Boulder, Colorado, USA.

James Henderson, Paola Merlo, Gabriele Musillo, and Ivan Titov. 2008. A latent variable model of synchronous parsing for syntactic and semantic dependencies. In *Proceedings of CoNLL-2008*, pages 178–182, Manchester, England, August.

Zheng Ping Jiang and Hwee Tou Ng. 2006. Semantic role labeling of nombank: A maximum entropy approach. In *Proceedings of EMNLP-2006*, pages 138–145, Sydney, Australia.

Richard Johansson and Pierre Nugues. 2008. Dependency-based syntactic－semantic analysis with propbank and nombank. In *Proceedings of CoNLL-2008*, page 183－187, Manchester, UK.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL-08: HLT*, pages 595–603, Columbus, Ohio, USA, June.

Peter Koomen, Vasin Punyakanok, Dan Roth, and Wen tau Yih. 2005. Generalized inference with multiple semantic role labeling systems. In *Proceedings of CoNLL-2005*, pages 181–184, Ann Arbor, Michigan, USA.

Chang Liu and Hwee Tou Ng. 2007. Learning predictive structures for semantic role labeling of nombank. In *Proceedings of ACL-2007*, pages 208–215, Prague, Czech.

Lluis Marquez, Mihai Surdeanu, Pere Comas, and Jordi Turmo. 2005. A robust combination strategy for semantic role labeling. In *Proceedings of HLT/EMNLP-2005*, page 644 – 651, Vancouver, Canada.

Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of CoNLL-X*, New York City, June.

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. The nombank project: An interim report. In *Proceedings of HLT/NAACL Workshop on Frontiers in Corpus Annotation*, pages 24–31, Boston, Massachusetts, USA, May 6.

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL-08: HLT*, pages 950–958, Columbus, Ohio, June.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James H. Martin, and Dan Jurafsky. 2004. Shallow semantic parsing using support vector machines. In *Proceedings of HLT/NAACL-2004*, pages 233–240, Boston, Massachusetts, USA.

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James H. Martin, and Daniel Jurafsky. 2005. Semantic role labeling using different syntactic views. In *Proceedings of ACL-2005*, pages 581–588, Ann Arbor, USA.

James Pustejovsky, Adam Meyers, Martha Palmer, and Massimo Poesio. 2005. Merging propbank, nombank, timebank, penn discourse treebank and coreference. In *Proceedings of the Workshop on Frontiers in Corpus Annotations II: Pie in the Sky*, pages 5–12, Ann Arbor, USA.

Mihai Surdeanu, Lluis Marquez, Xavier Carreras, and Pere R. Comas. 2007. Combination strategies for semantic role labeling. *Journal of Artificial Intelligence Research*, 29:105–151.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of CoNLL-2008*, pages 159–177, Manchester, UK.

Ivan Titov, James Henderson, Paola Merlo, and Gabriele Musillo. 2009. Online graph planarisation for synchronous parsing of semantic and syntactic dependencies. In *IJCAI-2009*, Pasadena, California, USA.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2005. Joint learning improves semantic role labeling. In *Proceedings of ACL-2005*, pages 589–596, Ann Arbor, USA.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of EMNLP-2004*, pages 88–94, Barcelona, Spain, July 25-26.

Nianwen Xue. 2006. Semantic role labeling of nominalized predicates in chinese. In *Proceedings of NAACL-2006*, pages 431–438, New York City, USA, June.

Hai Zhao and Chunyu Kit. 2008. Parsing syntactic and semantic dependencies with two single-stage maximum entropy models. In *Proceeding of CoNLL-2008*, pages 203–207, Manchester, UK.

Hai Zhao, Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009a. Multilingual dependency learning: Exploiting rich features for tagging syntactic and semantic dependencies. In *Proceedings of the 13th Conference on Computational Natural Language Learning (CoNLL-2009), June 4-5*, pages 61–66, Boulder, Colorado, USA.

Hai Zhao, Wenliang Chen, Chunyu Kit, and Guodong Zhou. 2009b. Multilingual dependency learning: A huge feature engineering method to semantic dependency parsing. In *Proceedings of CoNLL-2009*, pages 55–60, Boulder, Colorado, USA.