Semi-supervised Feature Transformation for Dependency Parsing

Wenliang Chen[†], **Min Zhang**[†*], **and Yue Zhang**[‡]
[†]School of Computer Science and Technology, Soochow University, China
[‡]Singapore University of Technology and Design, Singapore
{wlchen, mzhang}@suda.edu.cn
yue_zhang@sutd.edu.sg

Abstract

In current dependency parsing models, conventional features (i.e. base features) defined over surface words and part-of-speech tags in a relatively high-dimensional feature space may suffer from the data sparseness problem and thus exhibit less discriminative power on unseen data. In this paper, we propose a novel semi-supervised approach to addressing the problem by transforming the base features into high-level features (i.e. meta features) with the help of a large amount of automatically parsed data. The meta features are used together with base features in our final parser. Our studies indicate that our proposed approach is very effective in processing unseen data and features. Experiments on Chinese and English data sets show that the final parser achieves the best-reported accuracy on the Chinese data and comparable accuracy with the best known parsers on the English data.

1 Introduction

In recent years, supervised learning models have achieved lots of progress in the dependency parsing task, as can be found in the CoNLL shared tasks (Buchholz and Marsi, 2006; Nivre et al., 2007). The supervised models take annotated data as training data, utilize features defined over surface words, part-of-speech tags, and dependency trees, and learn the preference of features via adjusting feature weights.

In the supervised learning scenarios, many previous studies explore rich feature representation that leads to significant improvements. McDonald and Pereira (2006) and Carreras (2007) define second-order features over two adjacent arcs in second-order graph-based models. Koo and Collins (2010) use third-order features in a third-order graph-based model. Bohnet (2010) considers information of more surrounding words for the graph-based models, while Zhang and Nivre (2011) define a set of rich features including the word valency and the third-order context features for transition-based models. All these models utilize richer and more complex feature representations and achieve better performance than the earlier models that utilize the simpler features (McDonald et al., 2005; Yamada and Matsumoto, 2003; Nivre and Scholz, 2004). However, the richer feature representations result in a high-dimensional feature space. Features in such a space may suffer from the data sparseness problem and thus have less discriminative power on unseen data. If input sentences contain unknown features that are not included in training data, the parsers can usually give lower accuracy.

Several methods have been proposed to alleviate this problem by using large amounts of unannotated data, ranging from self-training and co-training (McClosky et al., 2006; Sagae and Tsujii, 2007) to more complex methods that collect statistical information from unannotated sentences and use them as additional features (Koo et al., 2008; Chen et al., 2009).

In this paper, we propose an alternative approach to semi-supervised dependency parsing via feature transformation (Ando and Zhang, 2005). More

[*]Corresponding author

specifically, we transform base features to a higher-level space. The base features defined over surface words, part-of-speech tags, and dependency trees are high dimensional and have been explored in the above previous studies. The higher-level features, which we call meta features, are low dimensional, and newly defined in this paper. The key idea behind is that we build connections between known and unknown base features via the meta features. From another viewpoint, we can also interpret the meta features as a way of doing feature smoothing.

Our feature transfer method is simpler than that of Ando and Zhang (2005), which is based on splitting the original problem into multiple auxiliary problems. In our approach, the base features are grouped and each group relates to a meta feature. In the first step, we use a baseline parser to parse a large amount of unannotated sentences. Then we collect the base features from the parse trees. The collected features are transformed into predefined discrete values via a transformation function. Based on the transformed values, we define a set of meta features. Finally, the meta features are incorporated directly into parsing models.

To demonstrate the effectiveness of the proposed approach, we apply it to the graph-based parsing models (McDonald and Nivre, 2007). We conduct experiments on the standard data split of the Penn English Treebank (Marcus et al., 1993) and the Chinese Treebank Version 5.1 (Xue et al., 2005). The results indicate that the approach significantly improves the accuracy. In summary, we make the following contributions:

- We define a simple yet useful transformation function to transform base features to meta features automatically. The meta features build connections between known and unknown base features, and relieve the data sparseness problem.
- Compared to the base features, the number of meta features is remarkably small.
- We build semi-supervised dependency parsers that achieve the best accuracy on the Chinese data and comparable accuracy with the best known systems on the English data.

The rest of this paper is organized as follows. Section 2 introduces the graph-based parsing model. Section 3 describes the meta features and meta parser. Section 4 describes the experiment settings and reports the experimental results on English and Chinese data sets. Section 5 discusses related work. Finally, in Section 6 we summarize the proposed approach.

2 Baseline parser

In this section, we introduce a graph-based parsing model proposed by McDonald et al. (2005) and build a baseline parser.

2.1 Graph-based parsing model

Given an input sentence, dependency parsing is to build a dependency tree. We define X as the set of possible input sentences, Y as the set of possible dependency trees, and $D = (x_1, y_1), ..., (x_i, y_i), ..., (x_n, y_n)$ as a training set of n pairs of $x_i \in X$ and $y_i \in Y$. A sentence is denoted by $x = (w_0, w_1, ..., w_i, ..., w_m)$, where w_0 is ROOT and does not depend on any other word and w_i refers to a word.

In the graph-based model, we define ordered pair $(w_i, w_j) \in y$ as a dependency relation in tree y from word w_i to word w_j (w_i is the head and w_j is the dependent), G_x as a graph that consists of a set of nodes $V_x = \{w_0, w_1, ..., w_i, ..., w_m\}$ and a set of arcs (edges) $E_x = \{(w_i, w_j) | i \neq j, w_i \in V_x, w_j \in (V_x - \{w_0\})\}$. The parsing model of McDonald et al. (2005) is to search for the maximum spanning tree (MST) in graph G_x. We denote $Y(G_x)$ as the set of all the subgraphs of G_x that are valid dependency trees (McDonald and Nivre, 2007) for sentence x.

We define the score of a dependency tree $y \in Y(G_x)$ to be the sum of the subgraph scores,

$$score(x, y) = \sum_{g \in y} score(x, g) \tag{1}$$

where g is a spanning subgraph of y, which can be a single arc or adjacent arcs. In this paper we assume the dependency tree to be a spanning projective tree. The model scores each subgraph using a linear representation. Then scoring function $score(x, g)$ is,

$$score(x, g) = \mathbf{f}(x, g) \cdot \mathbf{w} \tag{2}$$

where $\mathbf{f}(x, g)$ is a high-dimensional feature vector based on features defined over g and x and $\mathbf{w}$ refers to the weights for the features.

The maximum spanning tree is the highest scoring tree in $Y(G_x)$. The task of decoding algorithms in the parsing model for an input sentence x is to find y^*, where

$$\begin{aligned} y^* &= \underset{y \in Y(G_x)}{\arg\max}\, score(x, y) \\ &= \underset{y \in Y(G_x)}{\arg\max} \sum_{g \in y} score(x, g) \\ &= \underset{y \in Y(G_x)}{\arg\max} \sum_{g \in y} \mathbf{f}(x, g) \cdot \mathbf{w} \qquad (3) \end{aligned}$$

In our system, we use the decoding algorithm proposed by Carreras (2007), which is a second-order CKY-style algorithm (Eisner, 1996) and feature weights **w** are learned during training using the Margin Infused Relaxed Algorithm (MIRA) (Crammer and Singer, 2003; McDonald et al., 2005).

2.2 Base features

Previous studies have defined different sets of features for the graph-based parsing models, such as the first-order features defined in McDonald et al. (2005), the second-order parent-siblings features defined in McDonald and Pereira (2006), and the second-order parent-child-grandchild features defined in Carreras (2007). Bohnet (2010) explorers a richer set of features than the above sets. We further extend the features defined by Bohnet (2010) by introducing more lexical features as the base features. The base feature templates are listed in Table 1, where h, d refer to the head, the dependent respectively, c refers to d's sibling or child, b refers to the word between h and d, $+1$ (-1) refers to the next (previous) word, w and p refer to the surface word and part-of-speech tag respectively, $[wp]$ refers to the surface word or part-of-speech tag, $d(h, d)$ is the direction of the dependency relation between h and d, and $d(h, d, c)$ is the directions of the relation among h, d, and c. We generate the base features based on the above templates.

2.3 Baseline parser

We train a parser with the base features as the Baseline parser. We define $\mathbf{f}_b(x, g)$ as the base features and $\mathbf{w}_b$ as the corresponding weights. The scoring function becomes,

$$score(x, g) = \mathbf{f}_b(x, g) \cdot \mathbf{w}_b \qquad (4)$$

3 Meta features

In this section, we propose a semi-supervised approach to transform the features in the base feature space (F_B) to features in a higher-level space (F_M) with the following properties:

- The features in F_M are able to build connections between known and unknown features in F_B and therefore should be highly informative.
- The transformation should be learnable based on a labeled training set and an automatically parsed data set, and automatically computable for the test sentences.

The features in F_M are referred to as meta features. In order to perform the feature transformation, we choose to define a simple yet effective mapping function. Based on the mapped values, we define feature templates for generating the meta features. Finally, we build a new parser with the base and meta features.

3.1 Template-based mapping function

We define a template-based function for mapping the base features to predefined discrete values. We first put the base features into several groups and then perform mapping.

We have a set of base feature templates T_B. For each template $T_i \in T_B$, we can generate a set of base features F_i from dependency trees in the parsed data, which is automatically parsed by the Baseline parser. We collect the features and count their frequencies. The collected features are sorted in decreasing order of frequencies. The mapping function for a base feature f_b of F_i is defined as follows,

$$\Phi(f_b) = \begin{cases} H_i & \text{if } R(f_b) \leq \text{TOP10} \\ M_i & \text{if TOP10} < R(f_b) \leq \text{TOP30} \\ L_i & \text{if TOP30} < R(f_b) \\ O_i & \text{Others} \end{cases}$$

where $R(f_b)$ is the position number of f_b in the sorted list, "Others" is defined for the base features that are not included in the list, and TOP10 and TOP 30 refer to the position numbers of top 10% and top 30% respectively. The numbers, 10% and 30%, are tuned on the development sets in the experiments. For a base feature generated from template T_i, we have four possible values: H_i, M_i, L_i, and O_i. In

(a) First-order standard

h$_{[wp]}$, d$_{[wp]}$, d(h,d)
h$_{[wp]}$, d(h,d)
d$_w$, d$_p$, d(h,d)
d$_{[wp]}$, d(h,d)
h$_w$, h$_p$, d$_w$, d$_p$, d(h,d)
h$_p$, h$_w$, d$_p$, d(h,d)
h$_w$, d$_w$, d$_p$, d(h,d)
h$_w$, h$_p$, d$_{[wp]}$, d(h,d)

(b) First-order Linear

h$_p$, b$_p$, d$_p$, d(h,d)
h$_p$, h$_{+1p}$, d$_{-1p}$, d$_p$, d(h,d)
h$_{-1p}$, h$_p$, d$_{-1p}$, d$_p$, d(h,d)
h$_p$, h$_{+1p}$, d$_p$, d$_{+1p}$, d(h,d)
h$_{-1p}$, h$_p$, d$_p$, d$_{+1p}$, d(h,d)

(c) Second-order standard

h$_p$, d$_p$, c$_p$, d(h,d,c)
h$_w$, d$_w$, c$_w$, d(h,d,c)
h$_p$, c$_{[wp]}$, d(h,d,c)
d$_p$, c$_{[wp]}$, d(h,d,c)
h$_w$, c$_{[wp]}$, d(h,d,c)
d$_w$, c$_{[wp]}$, d(h,d,c)

(d) Second-order Linear

h$_{[wp]}$, h$_{+1[wp]}$, c$_{[wp]}$, d(h,d,c)
h$_{-1[wp]}$, h$_{[wp]}$, c$_{[wp]}$, d(h,d,c)
h$_{[wp]}$, c$_{-1[wp]}$, c$_{[wp]}$, d(h,d,c)
h$_{[wp]}$, c$_{[wp]}$, c$_{+1[wp]}$, d(h,d,c)
h$_{-1[wp]}$, h$_{[wp]}$, c$_{-1[wp]}$, c$_{[wp]}$, d(h,d,c)
h$_{[wp]}$, h$_{+1[wp]}$, c$_{-1[wp]}$, c$_{[wp]}$, d(h,d,c)
h$_{-1[wp]}$, h$_{[wp]}$, c$_{[wp]}$, c$_{+1[wp]}$, d(h,d,c)
h$_{[wp]}$, h$_{+1[wp]}$, c$_{[wp]}$, c$_{+1[wp]}$, d(h,d,c)
d$_{[wp]}$, d$_{+1[wp]}$, c$_{[wp]}$, d(h,d,c)
d$_{-1[wp]}$, d$_{[wp]}$, c$_{[wp]}$, d(h,d,c)
d$_{[wp]}$, c$_{-1[wp]}$, c$_{[wp]}$, d(h,d,c)
d$_{[wp]}$, c$_{[wp]}$, c$_{+1[wp]}$, d(h,d,c)
d$_{[wp]}$, d$_{+1[wp]}$, c$_{-1[wp]}$, c$_{[wp]}$, d(h,d,c)
d$_{[wp]}$, d$_{+1[wp]}$, c$_{[wp]}$, c$_{+1[wp]}$, d(h,d,c)
d$_{-1[wp]}$, d$_{[wp]}$, c$_{-1[wp]}$, c$_{[wp]}$, d(h,d,c)
d$_{-1[wp]}$, d$_{[wp]}$, c$_{[wp]}$, c$_{+1[wp]}$, d(h,d,c)

Table 1: Base feature templates

total, we have $4 \times N(T_B)$ possible values for all the base features, where $N(T_B)$ refers to the number of the base feature templates, which is usually small. We can obtain the mapped values of all the collected features via the mapping function.

3.2 Meta feature templates

Based on the mapped values, we define meta feature templates in F_M for dependency parsing. The meta feature templates are listed in Table 2, where f_b is a base feature of F_B, h_p refers to the part-of-speech tag of the head and h_w refers to the surface word of the head. Of the table, the first template uses the mapped value only, the second and third templates combine the value with the head information. The number of the meta features is relatively small. It has $4 \times N(T_B)$ for the first type, $4 \times N(T_B) \times N(POS)$ for the second type, and $4 \times N(T_B) \times N(WORD)$ for the third one, where $N(POS)$ refers to the number of part-of-speech tags, $N(WORD)$ refers to the number of words. We remove any feature related to the surface form if the word is not one of the Top-N most frequent words in the training data. We used N=1000 for the experiments for this paper. This method can reduce the size of the feature sets. The empirical statistics of the feature sizes at Section 4.2.2 shows that the size of meta features is only 1.2% of base features.

$[\Phi(f_b)]$
$[\Phi(f_b)], h_p$
$[\Phi(f_b)], h_w$

Table 2: Meta feature templates

3.3 Generating meta features

We use an example to demonstrate how to generate the meta features based on the meta feature templates in practice. Suppose that we have sentence "I ate the meat with a fork." and want to generate the meta features for the relation among "ate", "meat", and "with", where "ate" is the head, "meat" is the dependent, and "with" is the closest left sibling of "meat". Figure 1 shows the example.

We demonstrate the generating procedure using template T_k = "$h_w, d_w, c_w, d(h,d,c)$" (the second template of Table 1-(c)), which contains the surface forms of the head, the dependent, its sibling, and the directions of the dependencies among h, d, and c. We can have a base feature "ate, meat, with, RIGHTSIB", where "RIGHTSIB" refers to the parent-siblings structure with the right direction. In the auto-parsed data, this feature occurs 200 times and ranks between TOP10 and TOP30. Accord-

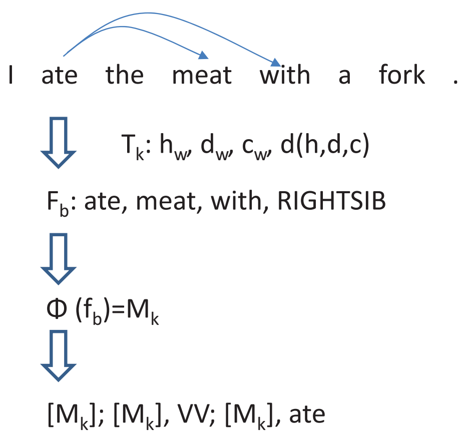
I ate the meat with a fork .
T_k: h_w, d_w, c_w, d(h,d,c)
F_b: ate, meat, with, RIGHTSIB
Φ (f_b)=M_k
[M_k]; [M_k], VV; [M_k], ate

Figure 1: An example of generating meta features

ing to the mapping function, we obtain the mapped value M_k. Finally, we have the three meta features "$[M_k]$", "$[M_k], VV$", and "$[M_k], ate$", where VV is the part-of-speech tag of word "ate". In this way, we can generate all the meta features for the graph-based model.

3.4 Meta parser

We combine the base features with the meta features by a new scoring function,

$$score(x, g) = \mathbf{f}_b(x, g) \cdot \mathbf{w}_b + \mathbf{f}_m(x, g) \cdot \mathbf{w}_m \quad (5)$$

where $\mathbf{f}_b(x, g)$ refers to the base features, $\mathbf{f}_m(x, g)$ refers to the meta features, and $\mathbf{w}_b$ and $\mathbf{w}_m$ are their corresponding weights respectively. The feature weights are learned during training using MIRA (Crammer and Singer, 2003; McDonald et al., 2005). Note that $\mathbf{w}_b$ is also retrained here.

We use the same decoding algorithm in the new parser as in the Baseline parser. The new parser is referred to as the meta parser.

4 Experiments

We evaluated the effect of the meta features for the graph-based parsers on English and Chinese data.

4.1 Experimental settings

In our experiments, we used the Penn Treebank (PTB) (Marcus et al., 1993) for English and the Chinese Treebank version 5.1 (CTB5) (Xue et al., 2005) for Chinese. The tool "Penn2Malt"[1] was used to convert the data into dependency structures with the English head rules of Yamada and Matsumoto (2003) and the Chinese head rules of Zhang and Clark (2008). We followed the standard data splits as shown in Table 3. Following the work of Koo et al. (2008), we used a tagger trained on training data to provide part-of-speech (POS) tags for the development and test sets, and used 10-way jackknifing to generate part-of-speech tags for the training set. We used the MXPOST (Ratnaparkhi, 1996) tagger for English and the CRF-based tagger for Chinese. We used gold standard segmentation in the CTB5. The data partition of Chinese were chosen to match previous work (Duan et al., 2007; Li et al., 2011; Hatori et al., 2011).

	train	dev	test
PTB (sections)	2-21	22	23
CTB5 (files)	001-815 1001-1136	886-931 1148-1151	816-885 1137-1147

Table 3: Standard data splits

For the unannotated data in English, we used the BLLIP WSJ corpus (Charniak et al., 2000) containing about 43 million words.[2] We used the MXPOST tagger trained on the training data to assign part-of-speech tags and used the Baseline parser to process the sentences of the Brown corpus. For the unannotated data in Chinese, we used the Xinhua portion of Chinese Gigaword[3] Version 2.0 (LDC2009T14) (Huang, 2009), which has approximately 311 million words. We used the MMA system (Kruengkrai et al., 2009) trained on the training data to perform word segmentation and POS tagging and used the Baseline parser to parse the sentences in the Gigaword data.

In collecting the base features, we removed the features which occur only once in the English data and less than four times in the Chinese data. The feature occurrences of one time and four times are based on the development data performance.

We measured the parser quality by the unlabeled attachment score (UAS), i.e., the percentage of to-

[1]http://w3.msi.vxu.se/~nivre/research/Penn2Malt.html

[2]We ensured that the text used for building the meta features did not include the sentences of the Penn Treebank.

[3]We excluded the sentences of the CTB data from the Gigaword data.

kens (excluding all punctuation tokens) with the correct HEAD. We also reported the scores on complete dependency trees evaluation (COMP).

4.2 Feature selection on development sets

We evaluated the parsers with different settings on the development sets to select the meta features.

4.2.1 Different models vs meta features

In this section, we investigated the effect of different types of meta features for the models trained on different sizes of training data on English.

There are too many base feature templates to test one by one. We divided the templates into several categories. Of Table 1, some templates are only related to part-of-speech tags (P), some are only related to surface words (W), and the others contain both part-of-speech tags and surfaces (M). Table 4 shows the categories, where numbers $[1-4]$ refer to the numbers of words involved in templates. For example, the templates of N3WM are related to three words and contain the templates of W and M. Based on different categories of base templates, we have different sets of meta features.[4]

Category	Example
N1P	$h_p, d(h,d)$
N1WM	$h_w, d(h,d)$; $h_w, h_p, d(h,d)$
N2P	$h_p, d_p, d(h,d)$
N2WM	$h_w, d_w, d(h,d)$; $h_w, d_p, d(h,d)$
N3P	$h_p, d_p, c_p, d(h,d,c)$
N3WM	$h_w, d_w, c_w, d(h,d,c)$; $d_w, d_{+1p}, c_p, d(h,d,c)$
N4P	$h_p, h_{+1p}, c_p, c_{+1p}, d(h,d,c)$
N4WM	$h_w, h_{+1w}, c_w, c_{+1w}, d(h,d,c)$; $h_w, h_{+1p}, c_p, c_{+1p}, d(h,d,c)$

Table 4: Categories of base feature templates

We randomly selected 1% and 10% of the sentences respectively from the training data. We trained the POS taggers and Baseline parsers on these small training data and used them to process the unannotated data. Then, we generated the meta features based on the newly auto-parsed data. The meta parsers were trained on the different subsets of the training data with different sets of meta features. Finally, we have three meta parsers: MP1, MP10, MPFULL, which were trained on 1%, 10% and 100% of the training data.

	MP1	MP10	MPFULL
Baseline	82.22	89.50	93.01
+N1P	82.42	89.48	93.08
+N1WM	82.80	89.42	93.19
+N2P	81.29	89.01	93.02
+N2WM	82.69	90.10	93.23
+N3P	83.32	89.73	93.05
+N3WM	84.47	90.75	93.80
+N4P	82.73	89.48	93.01
+N4WM	84.07	90.42	93.67
OURS	85.11	91.14	93.91

Table 5: Effect of different categories of meta features

Table 5 shows the results, where we add each category of Table 4 individually. From the table, we found that the meta features that are only related to part-of-speech tags did not always help, while the ones related to the surface words were very helpful. We also found that MP1 provided the largest relative improvement among the three settings. These suggested that the more sparse the base features were, the more effective the corresponding meta features were. Thus, we built the final parsers by adding the meta features of N1WM, N2WM, N3WM, and N4WM. The results showed that OURS achieved better performance than the systems with individual sets of meta features.

4.2.2 Different meta feature types

In Table 2, there are three types of meta feature templates. Here, the results of the parsers with different settings are shown in Table 6, where CORE refers to the first type, WithPOS refers to the second one, and WithWORD refers to the third one. The results showed that with all the types the parser (OURS) achieved the best. We also counted the numbers of the meta features. Only 327,864 (or 1.2%) features were added into OURS. Thus, we used all the three types of meta features in our final meta parsers.

[4]We also tested the settings of dividing WM into two subtypes: W and M. The results showed that both two sub-types provided positive results. To simplify, we merged W and M into one category WM.

System	NumOfFeat	UAS
Baseline	27,119,354	93.01
+CORE	+498	93.84
+WithPOS	+14,993	93.82
+WithWORD	+312,373	93.27
OURS	+327,864	93.91

Table 6: Numbers of meta features

4.3 Main results on test sets

We then evaluated the meta parsers on the English and Chinese test sets.

4.3.1 English

The results are shown in Table 7, where MetaParser refers to the meta parser. We found that the meta parser outperformed the baseline with an absolute improvement of 1.01 points (UAS). The improvement was significant in McNemar's Test (p $< 10^{-7}$).

	UAS	COMP
Baseline	92.76	48.05
MetaParser	93.77	51.36

Table 7: Main results on English

4.3.2 Chinese

	UAS	COMP
Baseline	81.01	29.71
MetaParser	83.08	32.21

Table 8: Main results on Chinese

The results are shown in Table 8. As in the experiment on English, the meta parser outperformed the baseline. We obtained an absolute improvement of 2.07 points (UAS). The improvement was significant in McNemar's Test (p $< 10^{-8}$).

In summary, Tables 7 and 8 convincingly show the effectiveness of our proposed approach.

4.4 Different sizes of unannotated data

Here, we considered the improvement relative to the sizes of the unannotated data used to generate the meta features. We randomly selected the 0.1%, 1%, and 10% of the sentences from the full data. Table 9 shows the results, where P0.1, P1, and P10 correspond to 0.1%, 1%, and 10% respectively. From the table, we found that the parsers obtained more benefits as we used more raw sentences. We also tried generating the meta features from the training data only, shown as TrainData in Table 9. However, the results shows that the parsers performed worse than the baselines. This is not surprising because only the known base features are included in the training data.

	English	Chinese
Baseline	92.76	81.01
TrainData	91.93	80.40
P0.1	92.82	81.58
P1	93.14	82.23
P10	93.48	82.81
FULL	93.77	83.08

Table 9: Effect of different sizes of auto-parsed data

4.5 Comparison with previous work

4.5.1 English

Table 10 shows the performance of the previous systems that were compared, where McDonald06 refers to the second-order parser of McDonald and Pereira (2006), Koo10 refers to the third-order parser with model1 of Koo and Collins (2010), Zhang11 refers to the parser of Zhang and Nivre (2011), Li12 refers to the unlabeled parser of Li et al. (2012), Koo08 refers to the parser of Koo et al. (2008), Suzuki09 refers to the parser of Suzuki et al. (2009), Chen09 refers to the parser of Chen et al. (2009), Zhou11 refers to the parser of Zhou et al. (2011), Suzuki11 refers to the parser of Suzuki et al. (2011), and Chen12 refers to the parser of Chen et al. (2012).

The results showed that our meta parser outperformed most of the previous systems and obtained the comparable accuracy with the best result of Suzuki11 (Suzuki et al., 2011) which combined the clustering-based word representations of Koo et al. (2008) and a condensed feature representation. However, our approach is much simpler than theirs and we believe that our meta parser can be further improved by combining their methods.

Type	System	UAS	COMP
Sup	McDonald06	91.5	
	Koo10	93.04	-
	Zhang11	92.9	48.0
	Li12	93.12	-
	Our Baseline	92.76	48.05
Semi	Koo08	93.16	
	Suzuki09	93.79	
	Chen09	93.16	47.15
	Zhou11	92.64	46.61
	Suzuki11	94.22	-
	Chen12	92.76	-
	MetaParser	93.77	51.36

Table 10: Relevant results for English. Sup denotes the supervised parsers, Semi denotes the parsers with semi-supervised methods.

4.5.2 Chinese

Table 11 shows the comparative results, where Li11 refers to the parser of Li et al. (2011), Hatori11 refers to the parser of Hatori et al. (2011), and Li12 refers to the unlabeled parser of Li et al. (2012). The reported scores on this data were produced by the supervised learning methods and our Baseline (supervised) parser provided the comparable accuracy. We found that the score of our meta parser for this data was the best reported so far and significantly higher than the previous scores. Note that we used the auto-assigned POS tags in the test set to match the above previous studies.

System	UAS	COMP
Li11	80.79	29.11
Hatori11	81.33	29.90
Li12	81.21	-
Our Baseline	81.01	29.71
MetaParser	83.08	32.21

Table 11: Relevant results for Chinese

4.6 Analysis

Here, we analyzed the effect of the meta features on the data sparseness problem.

We first checked the effect of unknown features on the parsing accuracy. We calculated the number of unknown features in each sentence and computed the average number per word. The average numbers were used to eliminate the influence of varied sentence sizes. We sorted the test sentences in increasing orders of these average numbers, and divided equally into five bins. BIN 1 is assigned the sentences with the smallest numbers and BIN 5 is with the largest ones. Figure 2 shows the average accuracy scores of the Baseline parsers against to the bins. From the figure, we found that for both two languages the Baseline parsers performed worse while the sentences contained more unknown features.

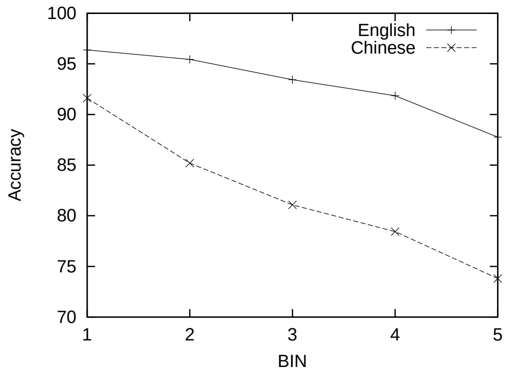

Figure 2: Accuracies relative to numbers of unknown features (average per word) by Baseline parsers

Then, we investigated the effect of the meta features. We calculated the average number of **active meta features** per word that were transformed from the unknown features for each sentence. We sorted the sentences in increasing order of the average numbers of active meta features and divided them into five bins. BIN 1 is assigned the sentences with the smallest numbers and BIN 5 is with the largest ones. Figures 3 and 4 show the results, where "Better" is for the sentences where the meta parsers provided better results than the baselines and "Worse" is for those where the meta parsers provided worse results. We found that the gap between "Better" and "Worse" became larger while the sentences contain more active meta features for the unknown features. The gap means performance improvement. This indicates that the meta features are very effective in processing the unknown features.

5 Related work

Our approach is to use unannotated data to generate the meta features to improve dependency parsing.

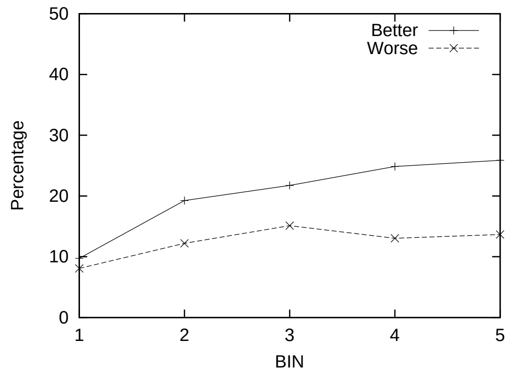

Figure 3: Improvement relative to numbers of active meta features on English (average per word)

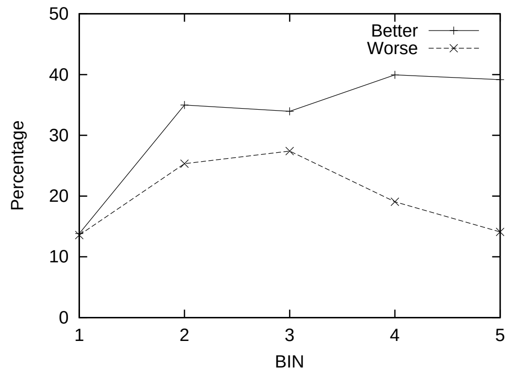

Figure 4: Improvement relative to numbers of active meta features on Chinese (average per word)

Several previous studies relevant to our approach have been conducted.

Koo et al. (2008) used a word clusters trained on a large amount of unannotated data and designed a set of new features based on the clusters for dependency parsing models. Chen et al. (2009) extracted subtree structures from a large amount of data and represented them as the additional features to improve dependency parsing. Suzuki et al. (2009) extended a Semi-supervised Structured Conditional Model (SS-SCM) of Suzuki and Isozaki (2008) to the dependency parsing problem and combined their method with the word clustering feature representation of Koo et al. (2008). Chen et al. (2012) proposed an approach to representing high-order features for graph-based dependency parsing models using a dependency language model and beam search. In future work, we may consider to combine their methods with ours to improve performance.

Several previous studies used co-training/self-training methods. McClosky et al. (2006) presented a self-training method combined with a reranking algorithm for constituency parsing. Sagae and Tsujii (2007) applied the standard co-training method for dependency parsing. In their approaches, some automatically parsed sentences were selected as new training data, which was used together with the original labeled data to retrain a new parser. We are able to use their approaches on top of the output of our parsers.

With regard to feature transformation, the work of Ando and Zhang (2005) is similar in spirit to our work. They studied semi-supervised text chunking by using a large projection matrix to map sparse base features into a small number of high level features. Their project matrix was trained by transforming the original problem into a large number of auxiliary problems, obtaining training data for the auxiliary problems by automatically labeling raw data and using alternating structure optimization to estimate the matrix across all auxiliary tasks. In comparison with their approach, our method is simpler in the sense that we do not request any intermediate step of splitting the prediction problem, and obtain meta features directly from self-annotated data. The training of our meta feature values is highly efficient, requiring the collection of simple statistics over base features from huge amount of data. Hence our method can potentially be useful to other tasks also.

6 Conclusion

In this paper, we have presented a simple but effective semi-supervised approach to learning the meta features from the auto-parsed data for dependency parsing. We build a meta parser by combining the meta features with the base features in a graph-based model. The experimental results show that the proposed approach significantly improves the accuracy. Our meta parser achieves comparable accuracy with the best known parsers on the English data (Penn English Treebank) and the best accuracy on the Chinese data (Chinese Treebank Version 5.1) so far. Further analysis indicate that the meta features are very effective in processing the unknown features. The idea described in this paper is general and can be applied to other NLP applications, such as part-of-speech tagging and Chinese word segmentation, in future work.

Acknowledgments

This study was started when Wenliang Chen and Min Zhang were members of the Department of Human Language Technology, Institute for Infocomm Research, Singapore. Wenliang Chen was funded partially by the National Science Foundation of China (61203314) and Yue Zhang was supported by MOE grant 2012-T2-2-163. We would also thank the anonymous reviewers for their detailed comments, which have helped us to improve the quality of this work.

References

R.K. Ando and T. Zhang. 2005. A high-performance semi-supervised learning method for text chunking. *ACL*.

Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 89–97, Beijing, China, August. Coling 2010 Organizing Committee.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of CoNLL-X*. SIGNLL.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 957–961, Prague, Czech Republic, June. Association for Computational Linguistics.

Eugene Charniak, Don Blaheta, Niyu Ge, Keith Hall, John Hale, and Mark Johnson. 2000. BLLIP 1987-89 WSJ Corpus Release 1, LDC2000T43. *Linguistic Data Consortium*.

Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009. Improving dependency parsing with subtrees from auto-parsed data. In *Proceedings of EMNLP 2009*, pages 570–579, Singapore, August.

Wenliang Chen, Min Zhang, and Haizhou Li. 2012. Utilizing dependency language models for graph-based dependency parsing models. In *Proceedings of ACL 2012*, Korea, July.

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *J. Mach. Learn. Res.*, 3:951–991.

Xiangyu Duan, Jun Zhao, and Bo Xu. 2007. Probabilistic models for action-based chinese dependency parsing. In *Proceedings of ECML/ECPPKDD*, Warsaw, Poland.

J. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of COLING1996*, pages 340–345.

Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2011. Incremental joint pos tagging and dependency parsing in chinese. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1216–1224, Chiang Mai, Thailand, November. Asian Federation of Natural Language Processing.

Chu-Ren Huang. 2009. Tagged Chinese Gigaword Version 2.0, LDC2009T14. *Linguistic Data Consortium*.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of ACL 2010*, pages 1–11, Uppsala, Sweden, July. Association for Computational Linguistics.

T. Koo, X. Carreras, and M. Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL-08: HLT*, Columbus, Ohio, June.

Canasai Kruengkrai, Kiyotaka Uchimoto, Jun'ichi Kazama, Yiou Wang, Kentaro Torisawa, and Hitoshi Isahara. 2009. An error-driven word-character hybrid model for joint Chinese word segmentation and POS tagging. In *Proceedings of ACL-IJCNLP2009*, pages 513–521, Suntec, Singapore, August. Association for Computational Linguistics.

Zhenghua Li, Min Zhang, Wanxiang Che, Ting Liu, Wenliang Chen, and Haizhou Li. 2011. Joint models for chinese pos tagging and dependency parsing. In *Proceedings of EMNLP 2011*, UK, July.

Zhenghua Li, Min Zhang, Wanxiang Che, and Ting Liu. 2012. A separately passive-aggressive training algorithm for joint pos tagging and dependency parsing. In *Proceedings of the 24rd International Conference on Computational Linguistics (Coling 2012)*, Mumbai, India. Coling 2012 Organizing Committee.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguisticss*, 19(2):313–330.

D. McClosky, E. Charniak, and M. Johnson. 2006. Reranking and self-training for parser adaptation. In *Proceedings of Coling-ACL*, pages 337–344.

R. McDonald and J. Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of EMNLP-CoNLL*, pages 122–131.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL 2006*, pages 81–88.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL 2005*, pages 91–98. Association for Computational Linguistics.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of English text. In *Proc. of the 20th Intern. Conf. on Computational Linguistics (COLING)*, pages 64–70.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of EMNLP 1996*, pages 133–142.

K. Sagae and J. Tsujii. 2007. Dependency parsing and domain adaptation with LR models and parser ensembles. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 1044–1050.

Jun Suzuki and Hideki Isozaki. 2008. Semi-supervised sequential labeling and segmentation using Giga-word scale unlabeled data. In *Proceedings of ACL-08: HLT*, pages 665–673, Columbus, Ohio, June. Association for Computational Linguistics.

Jun Suzuki, Hideki Isozaki, Xavier Carreras, and Michael Collins. 2009. An empirical study of semi-supervised structured conditional models for dependency parsing. In *Proceedings of EMNLP2009*, pages 551–560, Singapore, August. Association for Computational Linguistics.

Jun Suzuki, Hideki Isozaki, and Masaaki Nagata. 2011. Learning condensed feature representations from large unsupervised data sets for supervised learning. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 636–641, Portland, Oregon, USA, June. Association for Computational Linguistics.

Nianwen Xue, Fei Xia, Fu dong Chiou, and Martha Palmer. 2005. Building a Large Annotated Chinese Corpus: the Penn Chinese Treebank. *Journal of Natural Language Engineering*, 11(2):207–238.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT 2003*, pages 195–206.

Y. Zhang and S. Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of EMNLP 2008*, pages 562–571, Honolulu, Hawaii, October.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of ACL-HLT2011*, pages 188–193, Portland, Oregon, USA, June. Association for Computational Linguistics.

Guangyou Zhou, Jun Zhao, Kang Liu, and Li Cai. 2011. Exploiting web-derived selectional preference to improve statistical dependency parsing. In *Proceedings of ACL-HLT2011*, pages 1556–1565, Portland, Oregon, USA, June. Association for Computational Linguistics.