Utilizing Dependency Language Models for Graph-based Dependency Parsing Models

Wenliang Chen, Min Zhang*, and Haizhou Li
Human Language Technology, Institute for Infocomm Research, Singapore
{wechen, mzhang, hli}@i2r.a-star.edu.sg

Abstract

Most previous graph-based parsing models increase decoding complexity when they use high-order features due to exact-inference decoding. In this paper, we present an approach to enriching high-order feature representations for graph-based dependency parsing models using a dependency language model and beam search. The dependency language model is built on a large-amount of additional auto-parsed data that is processed by a baseline parser. Based on the dependency language model, we represent a set of features for the parsing model. Finally, the features are efficiently integrated into the parsing model during decoding using beam search. Our approach has two advantages. Firstly we utilize rich high-order features defined over a view of large scope and additional large raw corpus. Secondly our approach does not increase the decoding complexity. We evaluate the proposed approach on English and Chinese data. The experimental results show that our new parser achieves the best accuracy on the Chinese data and comparable accuracy with the best known systems on the English data.

1 Introduction

In recent years, there are many data-driven models that have been proposed for dependency parsing (McDonald and Nivre, 2007). Among them, graph-based dependency parsing models have achieved state-of-the-art performance for a wide range of languages as shown in recent CoNLL shared tasks (Buchholz and Marsi, 2006; Nivre et al., 2007). In the graph-based models, dependency parsing is treated as a structured prediction problem in which the graphs are usually represented as factored structures. The information of the factored structures decides the features that the models can utilize. There are several previous studies that exploit high-order features that lead to significant improvements.

McDonald et al. (2005) and Covington (2001) develop models that represent first-order features over a single arc in graphs. By extending the first-order model, McDonald and Pereira (2006) and Carreras (2007) exploit second-order features over two adjacent arcs in second-order models. Koo and Collins (2010) further propose a third-order model that uses third-order features. These models utilize higher-order feature representations and achieve better performance than the first-order models. But this achievement is at the cost of the higher decoding complexity, from $O(n^2)$ to $O(n^4)$, where n is the length of the input sentence. Thus, it is very hard to develop higher-order models further in this way.

How to enrich high-order feature representations without increasing the decoding complexity for graph-based models becomes a very challenging problem in the dependency parsing task. In this paper, we solve this issue by enriching the feature representations for a graph-based model using a dependency language model (DLM) (Shen et al., 2008). The N-gram DLM has the ability to predict the next child based on the N-1 immediate previous children and their head (Shen et al., 2008). The basic idea behind is that we use the DLM to evaluate whether a valid dependency tree (McDonald and Nivre, 2007)

*Corresponding author

Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, pages 213–222,
Jeju, Republic of Korea, 8-14 July 2012.

is well-formed from a view of large scope. The parsing model searches for the final dependency trees by considering the original scores and the scores of DLM.

In our approach, the DLM is built on a large amount of auto-parsed data, which is processed by an original first-order parser (McDonald et al., 2005). We represent the features based on the DLM. The DLM-based features can capture the N-gram information of the parent-children structures for the parsing model. Then, they are integrated directly in the decoding algorithms using beam-search. Our new parsing model can utilize rich high-order feature representations but without increasing the complexity.

To demonstrate the effectiveness of the proposed approach, we conduct experiments on English and Chinese data. The results indicate that the approach greatly improves the accuracy. In summary, we make the following contributions:

- We utilize the dependency language model to enhance the graph-based parsing model. The DLM-based features are integrated directly into the beam-search decoder.
- The new parsing model uses the rich high-order features defined over a view of large scope and and additional large raw corpus, but without increasing the decoding complexity.
- Our parser achieves the best accuracy on the Chinese data and comparable accuracy with the best known systems on the English data.

2 Dependency language model

Language models play a very important role for statistical machine translation (SMT). The standard N-gram based language model predicts the next word based on the $N-1$ immediate previous words. However, the traditional N-gram language model can not capture long-distance word relations. To overcome this problem, Shen et al. (2008) proposed a dependency language model (DLM) to exploit long-distance word relations for SMT. The N-gram DLM predicts the next child of a head based on the $N-1$ immediate previous children and the head itself. In this paper, we define a DLM, which is similar to the one of Shen et al. (2008), to score entire dependency trees.

An input sentence is denoted by $x = (x_0, x_1, ..., x_i, ..., x_n)$, where $x_0 = ROOT$ and does not depend on any other token in x and each token x_i refers to a word. Let y be a dependency tree for x and $H(y)$ be a set that includes the words that have at least one dependent. For each $x_h \in H(y)$, we have a dependency structure $D_h = (x_{Lk}, ...x_{L1}, x_h, x_{R1}...x_{Rm})$, where $x_{Lk}, ...x_{L1}$ are the children on the left side from the farthest to the nearest and $x_{R1}...x_{Rm}$ are the children on the right side from the nearest to the farthest. Probability $P(D_h)$ is defined as follows:

$$P(D_h) = P_L(D_h) \times P_R(D_h) \quad (1)$$

Here P_L and P_R are left and right side generative probabilities respectively. Suppose, we use a N-gram dependency language model. P_L is defined as follows:

$$\begin{aligned} P_L(D_h) \approx\ & P_{Lc}(x_{L1}|x_h) \\ & \times P_{Lc}(x_{L2}|x_{L1}, x_h) \\ & \times ... \\ & \times P_{Lc}(x_{Lk}|x_{L(k-1)}, ..., x_{L(k-N+1)}, x_h) \end{aligned} \quad (2)$$

where the approximation is based on the nth order Markov assumption. The right side probability is similar. For a dependency tree, we calculate the probability as follows:

$$P(y) = \prod_{x_h \in H(y)} P(D_h) \quad (3)$$

In this paper, we use a linear model to calculate the scores for the parsing models (defined in Section 3.1). Accordingly, we reform Equation 3. We define $\mathbf{f}_{DLM}$ as a high-dimensional feature representation which is based on arbitrary features of P_{Lc}, P_{Rc} and x. Then, the DLM score of tree y is in turn computed as the inner product of $\mathbf{f}^{DLM}$ with a corresponding weight vector $\mathbf{w}^{DLM}$.

$$score^{DLM}(y) = \mathbf{f}^{DLM} \cdot \mathbf{w}^{DLM} \quad (4)$$

3 Parsing with dependency language model

In this section, we propose a parsing model which includes the dependency language model by extending the model of McDonald et al. (2005).

3.1 Graph-based parsing model

The graph-based parsing model aims to search for the maximum spanning tree (MST) in a graph (McDonald et al., 2005). We write $(x_i, x_j) \in y$ if there is a dependency in tree y from word x_i to word x_j (x_i is the head and x_j is the dependent). A graph, denoted by G_x, consists of a set of nodes $V_x = \{x_0, x_1, ..., x_i, ..., x_n\}$ and a set of arcs (edges) $E_x = \{(x_i, x_j)|i \neq j, x_i \in V_x, x_j \in (V_x - x_0)\}$, where the nodes in V_x are the words in x. Let $T(G_x)$ be the set of all the subgraphs of G_x that are valid dependency trees (McDonald and Nivre, 2007) for sentence x.

The formulation defines the score of a dependency tree $y \in T(G_x)$ to be the sum of the edge scores,

$$s(x, y) = \sum_{g \in y} score(\mathbf{w}, x, g) \quad (5)$$

where g is a spanning subgraph of y. g can be a single dependency or adjacent dependencies. Then y is represented as a set of factors. The model scores each factor using a weight vector $\mathbf{w}$ that contains the weights for the features to be learned during training using the Margin Infused Relaxed Algorithm (MIRA) (Crammer and Singer, 2003; McDonald and Pereira, 2006). The scoring function is

$$score(\mathbf{w}, x, g) = \mathbf{f}(x, g) \cdot \mathbf{w} \quad (6)$$

where $\mathbf{f}(x, g)$ is a high-dimensional feature representation which is based on arbitrary features of g and x.

The parsing model finds a *maximum spanning tree* (MST), which is the highest scoring tree in $T(G_x)$. The task of the decoding algorithm for a given sentence x is to find y^*,

$$y^* = \arg\max_{y \in T(G_x)} s(x, y) = \arg\max_{y \in T(G_x)} \sum_{g \in y} score(\mathbf{w}, x, g)$$

3.2 Add DLM scores

In our approach, we consider the scores of the DLM when searching for the maximum spanning tree. Then for a given sentence x, we find y^*_{DLM},

$$y^*_{DLM} = \arg\max_{y \in T(G_x)} (\sum_{g \in y} score(\mathbf{w}, x, g) + score^{DLM}(y))$$

After adding the DLM scores, the new parsing model can capture richer information. Figure 1 illustrates the changes. In the original first-order parsing model, we only utilize the information of single arc (x_h, $x_{L(k-1)}$) for $x_{L(k-1)}$ as shown in Figure 1-(a). If we use 3-gram DLM, we can utilize the additional information of the two previous children (nearer to x_h than $x_{L(k-1)}$): $x_{L(k-2)}$ and $x_{L(k-3)}$ as shown in Figure 1-(b).

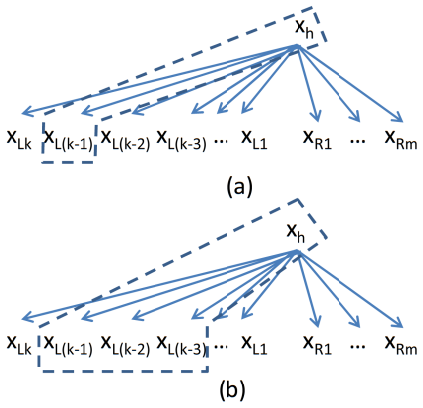

Figure 1: Adding the DLM scores to the parsing model

3.3 DLM-based feature templates

We define DLM-based features for $D_h = (x_{Lk}, ...x_{L1}, x_h, x_{R1}...x_{Rm})$. For each child x_{ch} on the left side, we have $P_{Lc}(x_{ch}|\text{HIS})$, where HIS refers to the $N-1$ immediate previous right children and head x_h. Similarly, we have $P_{Rc}(x_{ch}|\text{HIS})$ for each child on the right side. Let $P_u(x_{ch}|\text{HIS})$ ($P_u(ch)$ in short) be one of the above probabilities. We use the map function $\Phi(P_u(ch))$ to obtain the predefined discrete value (defined in Section 5.3). The feature templates are outlined in Table 1, where TYPE refers to one of the types:P_L or P_R, h_pos refers to the part-of-speech tag of x_h, h_word refers to the lexical form of x_h, ch_pos refers to the part-of-speech tag of x_{ch}, and ch_word refers to the lexical form of x_{ch}.

4 Decoding

In this section, we turn to the problem of adding the DLM in the decoding algorithm. We propose two ways: (1) Rescoring, in which we rescore the K-best list with the DLM-based features; (2) Intersect,

$< \Phi(P_u(ch)), \text{TYPE} >$
$< \Phi(P_u(ch)), \text{TYPE}, h\_pos >$
$< \Phi(P_u(ch)), \text{TYPE}, h\_word >$
$< \Phi(P_u(ch)), \text{TYPE}, ch\_pos >$
$< \Phi(P_u(ch)), \text{TYPE}, ch\_word >$
$< \Phi(P_u(ch)), \text{TYPE}, h\_pos, ch\_pos >$
$< \Phi(P_u(ch)), \text{TYPE}, h\_word, ch\_word >$

Table 1: DLM-based feature templates

in which we add the DLM-based features in the decoding algorithm directly.

4.1 Rescoring

We add the DLM-based features into the decoding procedure by using the rescoring technique used in (Shen et al., 2008). We can use an original parser to produce the K-best list. This method has the potential to be very fast. However, because the performance of this method is restricted to the K-best list, we may have to set K to a high number in order to find the best parsing tree (with DLM) or a tree acceptably close to the best (Shen et al., 2008).

4.2 Intersect

Then, we add the DLM-based features in the decoding algorithm directly. The DLM-based features are generated online during decoding.

For our parser, we use the decoding algorithm of McDonald et al. (2005). The algorithm was extensions of the parsing algorithm of (Eisner, 1996), which was a modified version of the CKY chart parsing algorithm. Here, we describe how to add the DLM-based features in the first-order algorithm. The second-order and higher-order algorithms can be extended by the similar way.

The parsing algorithm independently parses the left and right dependents of a word and combines them later. There are two types of chart items (McDonald and Pereira, 2006): 1) a *complete item* in which the words are unable to accept more dependents in a certain direction; and 2) an *incomplete item* in which the words can accept more dependents in a certain direction. In the algorithm, we create both types of chart items with two directions for all the word pairs in a given sentence. The direction of a dependency is from the head to the dependent. The right (left) direction indicates the dependent is on the right (left) side of the head. Larger chart items are created from pairs of smaller ones in a bottom-up style. In the following figures, complete items are represented by triangles and incomplete items are represented by trapezoids. Figure 2 illustrates the cubic parsing actions of the algorithm (Eisner, 1996) in the right direction, where s, r, and t refer to the start and end indices of the chart items. In Figure 2-(a), all the items on the left side are complete and the algorithm creates the incomplete item (trapezoid on the right side) of $s - t$. This action builds a dependency relation from s to t. In Figure 2-(b), the item of $s - r$ is incomplete and the item of $r - t$ is complete. Then the algorithm creates the complete item of $s - t$. In this action, all the children of r are generated. In Figure 2, the longer vertical edge in a triangle or a trapezoid corresponds to the subroot of the structure (spanning chart). For example, s is the subroot of the span $s - t$ in Figure 2-(a). For the left direction case, the actions are similar.

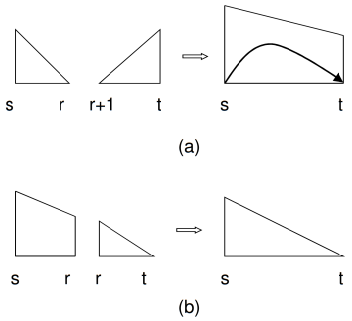

Figure 2: Cubic parsing actions of Eisner (Eisner, 1996)

Then, we add the DLM-based features into the parsing actions. Because the parsing algorithm is in the bottom-up style, the nearer children are generated earlier than the farther ones of the same head. Thus, we calculate the left or right side probability for a new child when a new dependency relation is built. For Figure 2-(a), we add the features of $P_{Rc}(x_t|HIS)$. Figure 3 shows the structure, where c_{Rs} refers to the current children (nearer than x_t) of x_s. In the figure, HIS includes c_{Rs} and x_s.

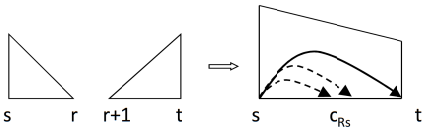

Figure 3: Add DLM-based features in cubic parsing

We use beam search to choose the one having the overall best score as the final parse, where K spans are built at each step (Zhang and Clark, 2008). At each step, we perform the parsing actions in the current beam and then choose the best K resulting spans for the next step. The time complexity of the new decoding algorithm is $O(Kn^3)$ while the original one is $O(n^3)$, where n is the length of the input sentence. With the rich feature set in Table 1, the running time of Intersect is longer than the time of Rescoring. But Intersect considers more combination of spans with the DLM-based features than Rescoring that is only given a K-best list.

5 Implementation Details

5.1 Baseline parser

We implement our parsers based on the MSTParser[1], a freely available implementation of the graph-based model proposed by (McDonald and Pereira, 2006). We train a first-order parser on the training data (described in Section 6.1) with the features defined in McDonald et al. (2005). We call this first-order parser Baseline parser.

5.2 Build dependency language models

We use a large amount of unannotated data to build the dependency language model. We first perform word segmentation (if needed) and part-of-speech tagging. After that, we obtain the word-segmented sentences with the part-of-speech tags. Then the sentences are parsed by the Baseline parser. Finally, we obtain the auto-parsed data.

Given the dependency trees, we estimate the probability distribution by relative frequency:

$$P_u(x_{ch}|\text{HIS}) = \frac{count(x_{ch}, \text{HIS})}{\sum_{x'_{ch}} count(x'_{ch}, \text{HIS})} \quad (7)$$

No smoothing is performed because we use the mapping function for the feature representations.

5.3 Mapping function

We can define different mapping functions for the feature representations. Here, we use a simple way. First, the probabilities are sorted in decreasing order. Let $No(P_u(ch))$ be the position number of $P_u(ch)$ in the sorted list. The mapping function is:

$$\Phi(P_u(ch)) = \begin{cases} PH & \text{if } No(P_u(ch)) \leq \text{TOP10} \\ PM & \text{if TOP10} < No(P_u(ch)) \leq \text{TOP30} \\ PL & \text{if TOP30} < No(P_u(ch)) \\ PO & \text{if } P_u(ch)) = 0 \end{cases}$$

where 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.

6 Experiments

We conducted experiments on English and Chinese data.

6.1 Data sets

For English, we used the Penn Treebank (Marcus et al., 1993) in our experiments. We created a standard data split: sections 2-21 for training, section 22 for development, and section 23 for testing. Tool "Penn2Malt"[2] was used to convert the data into dependency structures using a standard set of head rules (Yamada and Matsumoto, 2003). Following the work of (Koo et al., 2008), we used the MXPOST (Ratnaparkhi, 1996) tagger trained on training data to provide part-of-speech tags for the development and the test set, and used 10-way jackknifing to generate part-of-speech tags for the training set. For the unannotated data, we used the BLLIP corpus (Charniak et al., 2000) that contains about 43 million words of WSJ text.[3] We used the MXPOST tagger trained on training data to assign part-of-speech tags and used the Baseline parser to process the sentences of the BLLIP corpus.

For Chinese, we used the Chinese Treebank (CTB) version 4.0[4] in the experiments. We also used the "Penn2Malt" tool to convert the data and created a data split: files 1-270 and files 400-931 for training, files 271-300 for testing, and files 301-325 for development. We used gold standard segmentation and part-of-speech tags in the CTB. The data partition and part-of-speech settings were chosen to match previous work (Chen et al., 2008; Yu et al., 2008; Chen et al., 2009). For the unannotated data, we used the XIN_CMN portion of Chinese Gigaword[5] Version 2.0 (LDC2009T14) (Huang, 2009),

[1] http://mstparser.sourceforge.net

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

[3] We ensured that the text used for extracting subtrees did not include the sentences of the Penn Treebank.

[4] http://www.cis.upenn.edu/~chinese/.

[5] We excluded the sentences of the CTB data from the Gigaword data

which has approximately 311 million words whose segmentation and POS tags are given. We discarded the annotations due to the differences in annotation policy between CTB and this corpus. 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 all the sentences in the data.

6.2 Features for basic and enhanced parsers

The previous studies have defined four types of features: (FT1) the first-order features defined in McDonald et al. (2005), (FT2SB) the second-order parent-siblings features defined in McDonald and Pereira (2006), (FT2GC) the second-order parent-child-grandchild features defined in Carreras (2007), and (FT3) the third-order features defined in (Koo and Collins, 2010).

We used the first- and second-order parsers of the MSTParser as the basic parsers. Then we enhanced them with other higher-order features using beam-search. Table 2 shows the feature settings of the systems, where MST1/2 refers to the basic first-/second-order parser and MSTB1/2 refers to the enhanced first-/second-order parser. MSTB1 and MSTB2 used the same feature setting, but used different order models. This resulted in the difference of using FT2SB (beam-search in MSTB1 vs exact-inference in MSTB2). We used these four parsers as the Baselines in the experiments.

System	Features
MST1	(FT1)
MSTB1	(FT1)+(FT2SB+FT2GC+FT3)
MST2	(FT1+FT2SB)
MSTB2	(FT1+FT2SB)+(FT2GC+FT3)

Table 2: Baseline parsers

We measured the parser quality by the unlabeled attachment score (UAS), i.e., the percentage of tokens (excluding all punctuation tokens) with the correct HEAD. In the following experiments, we used "Inter" to refer to the parser with Intersect, and "Rescore" to refer to the parser with Rescoring.

6.3 Development experiments

Since the setting of K (for beam search) affects our parsers, we studied its influence on the development set for English. We added the DLM-based features to MST1. Figure 4 shows the UAS curves on the development set, where K is beam size for Intersect and K-best for Rescoring, the X-axis represents K, and the Y-axis represents the UAS scores. The parsing performance generally increased as the K increased. The parser with Intersect always outperformed the one with Rescoring.

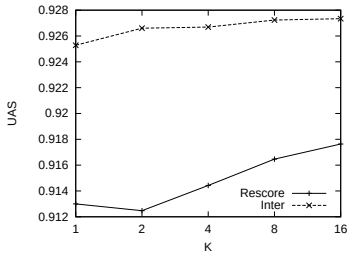

Figure 4: The influence of K on the development data

K	1	2	4	8	16
English	157.1	247.4	351.9	462.3	578.2

Table 3: The parsing times on the development set (seconds for all the sentences)

Table 3 shows the parsing times of Intersect on the development set for English. By comparing the curves of Figure 4, we can see that, while using larger K reduced the parsing speed, it improved the performance of our parsers. In the rest of the experiments, we set K=8 in order to obtain the high accuracy with reasonable speed and used Intersect to add the DLM-based features.

N	0	1	2	3	4
English	91.30	91.87	92.52	92.72	92.72
Chinese	87.36	87.96	89.33	89.92	90.40

Table 4: Effect of different N-gram DLMs

Then, we studied the effect of adding different N-gram DLMs to MST1. Table 4 shows the results. From the table, we found that the parsing performance roughly increased as the N increased. When N=3 and N=4, the parsers obtained the same scores for English. For Chinese, the parser obtained the best score when N=4. Note that the size of the Chinese unannotated data was larger than that of English. In the rest of the experiments, we used 3-gram for English and 4-gram for Chinese.

6.4 Main results on English data

We evaluated the systems on the testing data for English. The results are shown in Table 5, where -DLM refers to adding the DLM-based features to the Baselines. The parsers using the DLM-based features consistently outperformed the Baselines. For the basic models (MST1/2), we obtained absolute improvements of 0.94 and 0.63 points respectively. For the enhanced models (MSTB1/2), we found that there were 0.63 and 0.66 points improvements respectively. The improvements were significant in McNemar's Test ($p < 10^{-5}$)(Nivre et al., 2004).

Order1	UAS	Order2	UAS
MST1	90.95	MST2	91.71
MST-DLM1	91.89	MST-DLM2	92.34
MSTB1	91.92	MSTB2	92.10
MSTB-DLM1	92.55	MSTB-DLM2	92.76

Table 5: Main results for English

6.5 Main results on Chinese data

The results are shown in Table 6, where the abbreviations used are the same as those in Table 5. As in the English experiments, the parsers using the DLM-based features consistently outperformed the Baselines. For the basic models (MST1/2), we obtained absolute improvements of 4.28 and 3.51 points respectively. For the enhanced models (MSTB1/2), we got 3.00 and 2.93 points improvements respectively. We obtained large improvements on the Chinese data. The reasons may be that we use the very large amount of data and 4-gram DLM that captures high-order information. The improvements were significant in McNemar's Test ($p < 10^{-7}$).

Order1	UAS	Order2	UAS
MST1	86.38	MST2	88.11
MST-DLM1	90.66	MST-DLM2	91.62
MSTB1	88.38	MSTB2	88.66
MSTB-DLM1	91.38	MSTB-DLM2	91.59

Table 6: Main results for Chinese

6.6 Compare with previous work on English

Table 7 shows the performance of the graph-based systems that were compared, where McDonald06 refers to the second-order parser of McDonald and Pereira (2006), Koo08-standard refers to the second-order parser with the features defined in Koo et al. (2008), Koo10-model1 refers to the third-order parser with model1 of Koo and Collins (2010), Koo08-dep2c refers to the second-order parser with cluster-based features of (Koo et al., 2008), Suzuki09 refers to the parser of Suzuki et al. (2009), Chen09-ord2s refers to the second-order parser with subtree-based features of Chen et al. (2009), and Zhou11 refers to the second-order parser with web-derived selectional preference features of Zhou et al. (2011).

The results showed that our MSTB-DLM2 obtained the comparable accuracy with the previous state-of-the-art systems. Koo10-model1 (Koo and Collins, 2010) used the third-order features and achieved the best reported result among the supervised parsers. Suzuki2009 (Suzuki et al., 2009) reported the best reported result by combining a Semi-supervised Structured Conditional Model (Suzuki and Isozaki, 2008) with the method of (Koo et al., 2008). However, their decoding complexities were higher than ours and we believe that the performance of our parser can be further enhanced by integrating their methods with our parser.

Type	System	UAS	Cost
G	McDonald06	91.5	O(n^3)
	Koo08-standard	92.02	O(n^4)
	Koo10-model1	93.04	O(n^4)
S	Koo08-dep2c	93.16	O(n^4)
	Suzuki09	93.79	O(n^4)
	Chen09-ord2s	92.51	O(n^3)
	Zhou11	92.64	O(n^4)
D	MSTB-DLM2	92.76	O(Kn^3)

Table 7: Relevant results for English. G denotes the supervised graph-based parsers, S denotes the graph-based parsers with semi-supervised methods, D denotes our new parsers

6.7 Compare with previous work on Chinese

Table 8 shows the comparative results, where Chen08 refers to the parser of (Chen et al., 2008), Yu08 refers to the parser of (Yu et al., 2008), Zhao09 refers to the parser of (Zhao et al., 2009), and Chen09-ord2s refers to the second-order parser with subtree-based features of Chen et al. (2009). The results showed that our score for this data was the

best reported so far and significantly higher than the previous scores.

System	UAS
Chen08	86.52
Yu08	87.26
Zhao09	87.0
Chen09-ord2s	89.43
MSTB-DLM2	91.59

Table 8: Relevant results for Chinese

7 Analysis

Dependency parsers tend to perform worse on heads which have many children. Here, we studied the effect of DLM-based features for this structure. We calculated the number of children for each head and listed the accuracy changes for different numbers. We compared the MST-DLM1 and MST1 systems on the English data. The accuracy is the percentage of heads having all the correct children.

Figure 5 shows the results for English, where the X-axis represents the number of children, the Y-axis represents the accuracies, OURS refers to MST-DLM1, and Baseline refers to MST1. For example, for heads having two children, Baseline obtained 89.04% accuracy while OURS obtained 89.32%. From the figure, we found that OURS achieved better performance consistently in all cases and when the larger the number of children became, the more significant the performance improvement was.

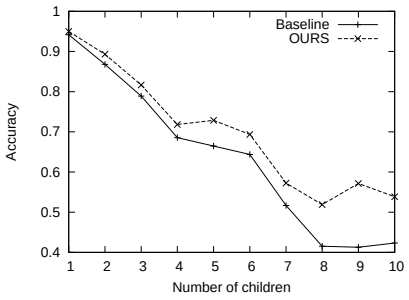

Figure 5: Improvement relative to numbers of children

8 Related work

Several previous studies related to our work have been conducted.

Koo et al. (2008) used a clustering algorithm to produce word clusters on a large amount of unannotated data and represented new features based on the clusters for dependency parsing models. Chen et al. (2009) proposed an approach that extracted partial tree structures from a large amount of data and used them as the additional features to improve dependency parsing. They approaches were still restricted in a small number of arcs in the graphs. Suzuki et al. (2009) presented a semi-supervised learning approach. They extended a Semi-supervised Structured Conditional Model (SS-SCM)(Suzuki and Isozaki, 2008) to the dependency parsing problem and combined their method with the approach of Koo et al. (2008). In future work, we may consider apply their methods on our parsers to improve further.

Another group of methods are the co-training/self-training techniques. McClosky et al. (2006) presented a self-training approach for phrase structure parsing. Sagae and Tsujii (2007) used the co-training technique to improve performance. First, two parsers were used to parse the sentences in unannotated data. Then they selected some sentences which have the same trees produced by those two parsers. They retrained a parser on newly parsed sentences and the original labeled data. We are able to use the output of our systems for co-training/self-training techniques.

9 Conclusion

We have presented an approach to utilizing the dependency language model to improve graph-based dependency parsing. We represent new features based on the dependency language model and integrate them in the decoding algorithm directly using beam-search. Our approach enriches the feature representations but without increasing the decoding complexity. When tested on both English and Chinese data, our parsers provided very competitive performance compared with the best systems on the English data and achieved the best performance on the Chinese data in the literature.

References

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, Daisuke Kawahara, Kiyotaka Uchimoto, Yujie Zhang, and Hitoshi Isahara. 2008. Dependency parsing with short dependency relations in unlabeled data. In *Proceedings of IJCNLP 2008*.

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.

Michael A. Covington. 2001. A dundamental algorithm for dependency parsing. In *Proceedings of the 39th Annual ACM Southeast Conference*, pages 95–102.

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

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

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.

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.

J. Nivre, J. Hall, and J. Nilsson. 2004. Memory-based dependency parsing. In *Proc. of CoNLL 2004*, pages 49–56.

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.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL-08: HLT*, pages 577–585, Columbus, Ohio, June. Association for Computational Linguistics.

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.

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

K. Yu, D. Kawahara, and S. Kurohashi. 2008. Chinese dependency parsing with large scale automatically constructed case structures. In *Proceedings of Coling 2008*, pages 1049–1056, Manchester, UK, August.

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.

Hai Zhao, Yan Song, Chunyu Kit, and Guodong Zhou. 2009. Cross language dependency parsing us-

ing a bilingual lexicon. In *Proceedings of ACL-IJCNLP2009*, pages 55–63, Suntec, Singapore, August. 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.