Muti-Path Shift-Reduce Parsing with Online Training

Wenbin Jiang, Hao Xiong, Qun Liu
Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, CAS
{jiangwenbin, xionghao, liuqun}@ict.ac.cn

Abstract: In this paper we describe an enhanced shift-reduce parsing method which differs from the traditional transition-based model in two ways: first, we maintain multiple transition paths after each transition step, in order to alleviate the serious risk of going astray for the only-one-path transition; second, we adopt the online training algorithm rather than the classical training-after-extraction method, to obtain more robust discriminativity of the classifier for transition decision. Experiments on the Tsinghua Chinese Treebank show that the enhanced model gains obvious improvement over the baseline. And in the CIPS evaluation task, a neat implementation without tricks could achieved nearly the state-of-the-art performance.

Keywords: Parsing, Transition-based, Online training

基于在线训练的多路移进归约句法分析模型

姜文斌, 熊皓, 刘群
中国科学院计算技术研究所智能信息处理重点实验室
{jiangwenbin, xionghao, liuqun}@ict.ac.cn

摘　要: 本文描述了一种增强型的移进归约句法分析模型。与传统的移进归约方法相比它有两个重要的改进：第一，该模型通过在每个转移步之后保留多个状态实现多路状态转移，较之单路径移进归约显著降低了状态转移的错误率；第二，取代传统的在抽取的实例上训练分类器的方法，该模型采用在线训练的方式调节分类器参数，较之离线训练方式考察了更多的分析状态从而习得更鲁棒的决策能力。在清华树库上的实验证实，这两个改进策略显著提高了分析性能。尽管我们的系统不借助任何细微技巧，但仍在CIPS评测中取得了相当可观的成绩。

关键词: 句法分析, 移进归约, 在线训练

1 Introduction

Most of the current state-of-the-art statistical parsers [Collins, 1999; Charniak, 2000; Petrov et al., 2006] are founded on the PCFG paradigm or its variants. Although well defined in the linguistic sense and usually giving encouraging performance, they suffer much from the high complexity of search space, from $O(n^3)$ to $O(n^5)$, leading to really lower processing speed especially for longer sentences.

Along with the popularity of discriminative methods in recent years, several classifier-based deterministic models are developed for dependency- and constituent parsing [Yamada and Matsumoto, 2003; Nivre et al., 2006; Sagae and Lavie, 2005]. Since these models perform parsing by making a series of shift-reduce decisions, they are also categorized as the transition-based method. Typically, in such models a stack is maintained to contain the parsed fragments so far, and a queue to contain the unprocessed tokens. A *shift* action moves the front of the

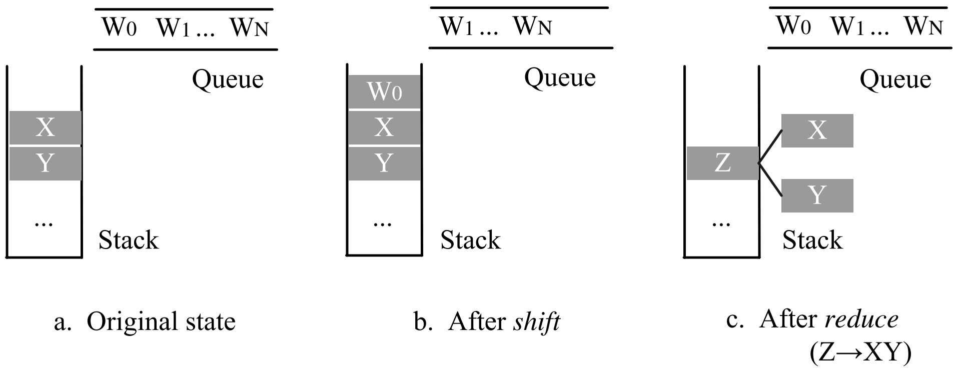

Figure 1: Classical shift-reduce parsing.

queue to the top of the stack, while a *reduce* actions merge several topmost elements of the stack into a larger fragment. A discriminative classifier, trained on a bunch of instances extracted from a treebank, is used to make the shift-reduce decisions. Previous experiments show that such models give very high processing speed that is linear to the length of input sentences, as well as acceptable performance that is not obvious worse than the state-of-the-art.

Besides the high processing speed, however, the greedy search procedure also brings higher risk of leading the transition to go astray. Since only one pair of stack and queue is maintained, only one transition path is traced during decoding, which means that each *state*, composed of a stack-queue pair and the current configuration, transfers in deterministic fashion to the next state. This indicates that any decision error will leads to a wrong parse. After adapting the shift-reduce paradigm to constituent parsing [Sagae and Lavie, 2005], Sagae and Lavie [2006] further proposed a best-first search strategy to balance accuracy and speed, which uses a prior heap to contain many possible states and each time expands the topmost state with the highest probability. Although achieving obvious performance improvement at the cost of decoding slowdown, this strategy is complicated and has a very small room for further improvement.

In this paper, we describe a more simpler improvement strategy, multi-path shift-reduce parsing, which maintains multiple transition paths during decoding by retaining several best derived states after each expansion. It can be seen as a discriminative version of the Generalized-LR parsing [Tomita, 1990], and is more applicable than the best-first strategy for the downstream NLP applications, such as n-best or forest generation for machine translation. In addition, while training the classifier we adopt the online training algorithm which updates parameters immediately at each decision error. Compared with previous training-after-extraction method which trains the classifier on the instances extracted from the treebank, online training captures more robust capability to escape from wrong transition paths. Experiments on the Tsinghua Chinese Treebank show that these two improvement strategies bring obvious performance improvement. And in the CIPS evaluation task, a neat implementation without tricks could achieved nearly the state-of-the-art performance.

In the rest of the paper, we first present the classical shift-reduce parsing algorithm (section 2) and its multi-path variant (section 3). After discussing the online training algorithm (section 4), we show the experiments (section 5).

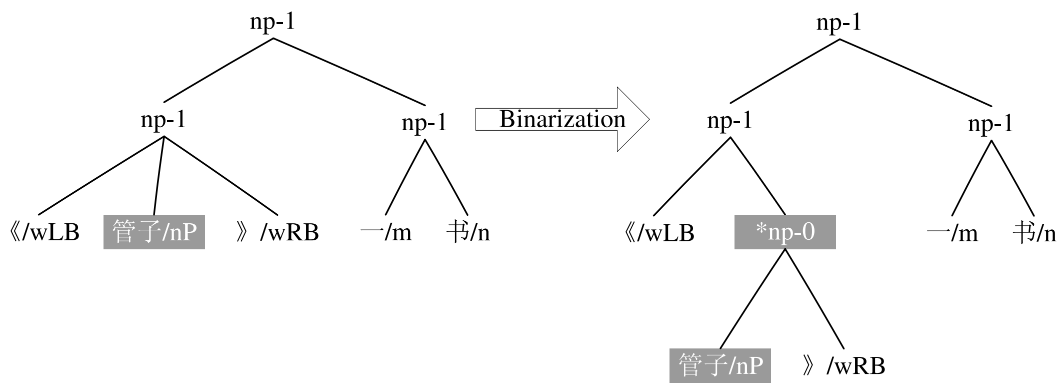

Figure 2: Binarization for a sub-tree from TCT.

2 Shift-Reduce Parsing

The idea of shift-reduce parsing is borrowed from compiler theory. It has been applied to constituency parsing, for example by Sagae and Lavie [2006]. Given an input sentence, it performs a left-to-right scan, and at each step chooses the best action according to the predication of the classifier. The action can be *shift* or one of the $|\mathbf{NT}|$ *reduce* actions. Here $|\mathbf{NT}|$ is the amount of the non-terminals, and each *reduce* corresponds to a non-terminal. The *shift* action shifts the current word onto the stack, while a *reduce* action merges several items on the top of the stack and then replaces them with their combination. Figure 1 shows an example of *shift* and *reduce* operations where the *reduce* action is *reduce-Z*.

To reduce the computational complexity, we follow the tradition of previous works [Johnson, 1998; Sagae and Lavie, 2006] to perform binarization to the treebank before training, and unbinarization to the parses after decoding. The purpose of binarization is to decompose the productions with more than two children, which can be solved by inserting pseudo internal non-terminals to such productions. To maintain the applicability of head-driven probabilistic training or discriminative training with head lexicon features, the treebank must be head-annotated, and the newly-created internal non-terminals must have the same head lexicon as the original productions. Each pseudo non-terminals are marked with a special symbol, so as to enable the unbinarization to the binarized parses after decoding. Figure 2 gives an example of binarization.

Another transformation we perform to the treebank is reduction of single-branches. Single-branch is a necessary of Chomsky's grammatical theory, but it brings a trouble to the decoding, where some mechanism must be established to terminate the derivation of single nodes. Especially in the shift-reduce paradigm, we have to introduce a new kind of operations which reduce only one node into another. The purpose of single-branch reduction is to decrease the operation set so that only binary reduce operations are sufficient for decoding. We use an example in Figure 3 to explain the procedure of single-branch reduction. The original tree is on the left, it has a single-branch composed of two nodes which are marked by shaded frames. The reduction operation replaces these two nodes by their combination, and labels the combined node with the combination of the non-terminals of the original two nodes.

The treebank consists only binary branch nodes after binarization for multi-branch nodes

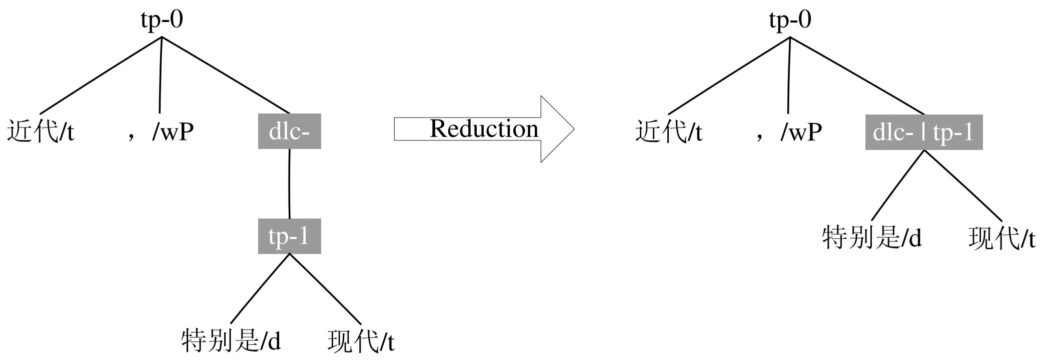

Figure 3: Single-branch reduction for a tree from TCT.

Algorithm 1 Multi-path shift-reduce parsing.

```
1:  Input: POS-tagged word sequence $x$
2:  $(ini \cdot stack, ini \cdot queue) \leftarrow (\emptyset, x)$
3:  insert $ini$ into $\mathbf{V}$
4:  for $step \leftarrow 1 .. 2|x| - 1$ do
5:      for each $state$ in $\mathbf{V}$ do
6:          for each $action$ that can be applied to $state$ do
7:              $next \leftarrow$ apply $action$ to $state$
8:              insert $next$ into $\mathbf{BUF}$
9:      $\mathbf{V} \leftarrow K$ best states of $\mathbf{BUF}$
10: Output: the tree derived from the best state in $\mathbf{V}$
```

and reduction for single-branches. Therefore, only a *shift* action and a series of binary *reduce* actions are needed for decoding. We omit the decoding algorithm here since the classical shift-reduce parsing is simple and well-known.

3 Multi-Path Shift-Reduce

We use an intuitive strategy, beam-search, to enhance the traditional shift-reduce parsing. Rather than maintaining one pair of stack and queue and performing deterministic shifting or reducing, a beam-search-style decoder resorts to several pairs of stack and queue to retain multiple transition paths. At each transition step, a series of current states are maintained by the stacks and queues. To obtain the series of following states, all the current states emit their possible derivations by conducting the *shfit* or *reduce* operations given by the classifier, and then the top best ones are selected out of all these derivations. Obviously, the total bunches of transitions for multi-path shift-reduce parsing is $2N - 1$, where N represents the sentence length. Suppose K best states are retained at each transition, the time complexity of the multi-path shift-reduce parser is then $O(KN)$.

Algorithm 1 depicts the procedure of multi-path shift-reduce parsing. The array $\mathbf{V}$ is used to contain the series of best states. Line $4 - 9$ perform the $2N - 1$ bunches of transitions. Line 9 chooses the K best candidates as the series of following states.

Algorithm 2 Online training for multi-path shift-reduce parsing.

```
1:  Input: Training examples $(x_i, y_i)$
2:  $\vec{\alpha} \leftarrow \mathbf{0}$
3:  for $t \leftarrow 1 .. T$ do
4:      for $i \leftarrow 1 .. N$ do
5:          $(ini \cdot stack, ini \cdot queue) \leftarrow (\emptyset, x_i)$
6:          insert $ini$ into $\mathbf{V}$
7:          for $step \leftarrow 1 .. 2|x_i| - 1$ do
8:              for each $state$ in $\mathbf{V}$ do
9:                  for each $action$ that can be applied to $state$ do
10:                     $next \leftarrow$ apply $action$ to $state$ according to $\vec{\alpha}$
11:                     insert $next$ into $\mathbf{BUF}$
12:             $\mathbf{V} \leftarrow K$ best states of $\mathbf{BUF}$
13:             if the oracle next state determined by $y_i$ falls out of $\mathbf{V}$ then
14:                 update $\vec{\alpha}$
15:                 break
16: Output: Parameters $\vec{\alpha}$
```

4 Online Training

A classical shift-reduce parser is trained on instances extracted from a treebank, where each instance is composed of a transition type and a set of features. Resort to a stack and a queue, an extractor simulates the shift-reduce parsing procedure for each tree in the transformed treebank, and outputs the instance at each transition. A batch-processed classifier is then trained on the instance set, and used to determine the transition choice during parsing.

Different from the batched, training-after-extraction manner for training a traditional shift-reduce parser, the online training for multi-path parsing decodes the sentences in the treebank one by one, and updates related parameters when the oracle next state falls out of the candidate list generated according to the current parameters. Compared with the batched training, the online training procedure usually costs more time since several iterations of decoding should be conducted across the training corpus. However, just thanks to the nearly exhaustive attempting and updating, online training achieves more robust discriminativity of escaping from wrong transition paths.

We describe the training procedure formally in Algorithm 2, where a simple perceptron algorithm [Collins, 2002] is adopted for parameter tuning. In each training instance (x_i, y_i), x_i is a token sequence and y_i is the constituent tree over it. The cascaded loop in line 3-4 iterates for T iterations across the N instances in the training corpus. For each instance, the loop in line 7 performs multi-path shift-reduce decoding for x_i according to the current parameters $\vec{\alpha}$. At any of the $2|x_i| - 1$ steps during decoding, it terminates the decoding procedure and updates the related parameters if the oracle next state determined by y_i falls out of the candidate list $\mathbf{V}$. To alleviate overfitting, the "averaged parameters" strategy is used in this algorithm. All the TN parameter sets, each of which is exported after each decoding, are averaged to obtain smoothed parameters. The iteration count T is chosen to maximize the performance on the developing set.

System	**Validating F1%**
offline+single	82.7
online+single	83.4
offline+multiple	85.3
online+multiple	**88.1**

Table 1: Performances of the series of parsers. *offline* and *online* respectively indicate the offline training and the online training, while *single* and *multiple* indicate the single-path transition and the multi-path transition. For the multi-path shift-reduce parser, we simply set the beam as 32.

Setting	**Headed F1%**
online+multiple	80.3
online+multiple+post	**84.0**

Table 2: Performance comparation between systems before and after post-process with hand-written head-recovery rules.

5 Experiments

We perform experiments on the training portion of the Tsinghua treebank provided by the organizers. Instances containing only one word are deleted since they don't provide any syntax information. Out of the filtered corpus, 500 trees are randomly extracted as the development set and another 500 as the validating set, with the remaining as the training set. Before the binarization, an additional operation should be performed to select one head for the spans with multiple heads, to facilitate the design of features with head information. In all experiments we simply choose the rightmost heads for such spans, and use some simple hand-written rules to recover the eliminated heads after parsing. As the development set and the validating set both account for a small proportion, the parsers trained on this training set are also used to parse the final test set. With the simple averaged perceptron algorithm a series of parsers are built, all of which are offline trained/online trained single-path/multi-path shift-reduce parsers.

Table 1 shows the performances of the four systems on the validating set. Offline+single is a simple reimplement version of [Sagae and Lavie, 2006] but with an offline perceptron algorithm for traning. Contrast to offline+single, online+single uses the online training algorithm described in section 4 and achieves a little improvement. Instead of maintaining only a single transition path after each expansion, multi-path shift-reduce parsing maintains multiple transition paths during decoding and, consequently, captures more capability to escape from wrong transition paths. This is validated by the performances of the X+multiple systems, which are significantly higher than the X-singles.

By observing the treebank, we design some linguistic rules to recovery the head words eliminated in the preprocessing. For example: from *vp-01(vp vp)* , we can capture the following rule:

$$childsize{=}{=}2\&\&child[0]{=}{=}child[1]\&\&child[0][0]{=}{=}\text{'}v\text{'} \Rightarrow head\ word{=}\text{"}01\text{"}$$

In addition to linguistic rules, some special rules were learned by statistic approach. Head-word which appears more than half of its children’ s occurrences was added into the rule table. In testing, we firstly used heuristic rules and then searched rule table for adjusting the head-word. Result showed in Table 2 indicates that the post-process achieves significant improvement over baseline in head-word evaluation.

Acknowledgements

We are grateful to the hard work of evaluation organizers in Tsinghua University and Northeastern University, and especially, we show our special thanks to Yumei Li in Tsinghua University for her generous help. This work was supported by National Natural Science Foundation of China, Contracts 60603095 and 60736014, and 863 State Key Project No. 2006AA010108.

References

[Charniak2000] Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL*.

[Collins1999] Michael Collins. 1999. Head-driven statistical models for natural language parsing. In *Ph.D. Disseration*.

[Collins2002] Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the EMNLP*, pages 1–8, Philadelphia, USA.

[Johnson1998] Mark Johnson. 1998. Pcfg models of linguistic tree representations. In *Computational Linguistics*.

[Nivre et al.2006] Joakim Nivre, Johan Hall, Jens Nilsson, Gulsen Eryigit, and Svetoslav Marinov. 2006. Labeled pseudoprojective dependency parsing with support vector machines. In *Proceedings of CoNLL*, pages 221–225.

[Petrov et al.2006] Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of ACL*.

[Sagae and Lavie2005] Kenji Sagae and Alon Lavie. 2005. A classifier-based parser with linear run-time complexity. In *Proceedings of IWPT*.

[Sagae and Lavie2006] Kenji Sagae and Alon Lavie. 2006. A best-first probabilistic shift-reduce parser. In *Proceedings of COLING/ACL*.

[Tomita1990] Masaru Tomita. 1990. The generalized lr parser/compiler - version 8.4. In *Proceedings of COLING*.

[Yamada and Matsumoto2003] H Yamada and Y Matsumoto. 2003. Statistical dependency analysis using support vector machines. In *Proceedings of IWPT*.