Experiment report for Shift-Reduce Parsing System

# Recall the current status of shift-reduce parsing system

## Feature teamplates which has been used.

There two feature templates which is used in our system: (Table 1 has show the notation used in our feature template)

* Base features (1): Zhang Yue’s baseline template. (Table 2)
* Lesser features (2): Our simpler feature template which is based on “less grammar, more features” paper.

Table 1. List of notation

|  |  |
| --- | --- |
| s0 | first node on stack |
| s1 | second node on stack |
| s2 | third node on stack |
| s3 | fourth node on stack |
| N0 | first word (node) on queue |
| N1 | second word (node) on queue |
| N2 | third word (node) on queue |
| N3 | fourth word (node) on queue |
| s0l | left child of s0 (if exist) |
| s0r | right child of s0 (if exist) |
| s0u | unary child of s0 (if exist) |
| s1l | left child of s1 (if exist) |
| s1r | right child of s1 (if exist) |
| s1u | unary child of s1 (if exist) |
| Xt | head PoS of node X |
| Xw | head word of node X |
| Xc | Constituent tag of node X |
| Xfw | first word within the span of node X |
| Xft | first PoS within the span of node X |
| Xlsw | last word within the span of node X |
| Xlst | last PoS within the span of node X |
| Xlen | the span length of node X |
| Xshape | the span shape of node X |
| Xaw | the word after span of node X |
| Xat | the PoS after span of node X |
| Xbw | the word before span of node X |
| Xbt | the PoS before span of node X |

Table 2. List of Base features

|  |  |  |
| --- | --- | --- |
| **Id** | **Feature** | **Description** |
|  | s0t\_s0c | s0: head PoS + constituent tag |
|  | s0w\_s0c | s0: head word + constituent tag |
|  | s1t\_s1c | s1: head PoS + constituent tag |
|  | s1w\_s1c | s1: head word + constituent tag |
|  | s2t\_s2c | s2: head PoS + constituent tag |
|  | s2w\_s2c | s2: head word + constituent tag |
|  | s3t\_s3c | s3: head PoS + constituent tag |
|  | s3w\_s3c | s3: head word + constituent tag |
|  | N0w\_N0t | N0: word + PoS |
|  | N1w\_N1t | N1: word + PoS |
|  | N2w\_N2t | N2: word + PoS |
|  | N3w\_N3t | N3: word + PoS |
|  | s0lw\_s0lc | s0l: head word + constituent tag |
|  | s0rw\_s0rc | s0r: head word + constituent tag |
|  | s0uw\_s0uc | s0u: head word + constituent tag |
|  | s1lw\_s1lc | s1l: head word + constituent tag |
|  | s1rw\_s1rc | s1r: head word + constituent tag |
|  | s1uw\_s1uc | s1u: head word + constituent tag |
|  | s0w\_s1w | s0: head word + s1: head word |
|  | s0w\_s1c | s0: head word + s1: constituent tag |
|  | s0c\_s1w | s0: constituent tag + s1: head word |
|  | s0c\_s1c | s0: constituent tag + s1: constituent tag |
|  | s0w\_N0w | s0: head word + N0: word |
|  | s0w\_N0t | s0: head word + N0: PoS |
|  | s0c\_N0w | s0: constituent tag + N0: word |
|  | s0c\_N0t | s0: constituent tag + N0: PoS |
|  | N0w\_N1w | N0: word + N1: word |
|  | N0w\_N1t | N0: word + N1: PoS |
|  | N0t\_N1w | N0: PoS + N1: word |
|  | N0t\_N1t | N0: PoS + N1: PoS |
|  | s1w\_N0w | s1: head word + N0: word |
|  | s1w\_N0t | s1: head word + N0: PoS |
|  | s1c\_N0w | s1: constituent tag + N0: word |
|  | s1c\_N0t | s1: constituent tag + N0: PoS |
|  | s0c\_s1c\_s2c | s0: constituent tag + s1: constituent tag + s2: constituent tag |
|  | s0w\_s1c\_s2c | s0: head word + s1: constituent tag + s2: constituent tag |
|  | s0c\_s1w\_s2c | s0: constituent tag + s1: head word + s2: constituent tag |
|  | s0c\_s1c\_s2w | s0: constituent tag + s1: constituent tag + s2: head word |
|  | s0c\_s1c\_N0t | s0: constituent tag + s1: constituent tag + N0: PoS |
|  | s0w\_s1c\_N0t | s0: head word + s1: constituent tag + N0: PoS |
|  | s0c\_s1w\_N0t | s0: constituent tag + s1: head word + N0: PoS |
|  | s0c\_s1c\_N0w | s0: constituent tag + s1: constituent tag + N0: word |

Table 3. List of lesser features

|  |  |  |
| --- | --- | --- |
| Id | Feature | Description |
|  | N0w\_N0t | N0: word + PoS |
|  | N1w\_N1t | N1: word + PoS |
|  | N2w\_N2t | N2: word + PoS |
|  | N3w\_N3t | N3: word + PoS |
|  | s0c\_s0ft | s0: constituent tag + first span PoS |
|  | s0c\_s0fw | s0: constituent tag + first span word |
|  | s0c\_s0lst | s0: constituent tag + last span PoS |
|  | s0c\_s0lsw | s0: constituent tag + last span word |
|  | s0c\_s0at | s0: constituent tag + after span PoS |
|  | s0c\_s0aw | s0: constituent tag + after span word |

|  |  |  |
| --- | --- | --- |
|  | s0c\_s0bt | s0: constituent tag + before span PoS |
|  | s0c\_s0bw | s0: constituent tag + before span word |
|  | s0c\_s0len | s0: constituent tag + span length |
|  | s1c\_s1ft | s1: constituent tag + first span PoS |
|  | s1c\_s1fw | s1: constituent tag + first span word |
|  | s1c\_s1lst | s1: constituent tag + last span PoS |
|  | s1c\_s1lsw | s1: constituent tag + last span word |
|  | s1c\_s1at | s1: constituent tag + after span PoS |
|  | s1c\_s1aw | s1: constituent tag + after span word |
|  | s1c\_s1bt | s1: constituent tag + before span PoS |
|  | s1c\_s1bw | s1: constituent tag + before span word |
|  | s1c\_s1len | s1: constituent tag + span length |
|  | s0c\_s0Shape | s0: constituent tag + span shape |
|  | s1c\_s1Shape | s1: constituent tag + span shape |
|  | s0c\_s0ft\_s0lst | s0: constituent tag + first span PoS + last span word |
|  | s0c\_s0fw\_s0lsw | s0: constituent tag + first span word + last span word |
|  | s1c\_s1ft\_s1lst | s1: constituent tag + first span PoS + last span PoS |
|  | s1c\_s1fw\_s1lsw | s1: constituent tag + first span word + last span word |
|  | s0c\_s0ft\_s0len | s0: constituent tag + first span PoS + span length |
|  | s0c\_s0fw\_s0len | s0: constituent tag + first span word + span length |
|  | s0c\_s0lst\_s0len | s0: constituent tag + last span PoS + span length |
|  | s0c\_s0lsw\_s0len | s0: constituent tag + last span word + span length |
|  | s1c\_s1ft\_s1len | s1: constituent tag + first span PoS + span length |
|  | s1c\_s1fw\_s1len | s1: constituent tag + first span word + span length |
|  | s1c\_s1lst\_s1len | s1: constituent tag + last span PoS + span length |
|  | s1c\_s1lsw\_s1len | s1: constituent tag + last span word + span length |
|  | s0c\_s0ft\_s0lst\_s0len | s0: constituent tag + first span PoS + last span PoS + span length |
|  | s0c\_s0ft\_s0lsw\_s0len | s0: constituent tag + first span PoS + last span word + span length |
|  | s0c\_s0fw\_s0lst\_s0len | s0: constituent tag + first span word + last span PoS + span length |
|  | s0c\_s0fw\_s0lsw\_s0len | s0: constitutent tag + first span word + last span word + span length |
|  | s1c\_s1ft\_s1lst\_s1len | s1: constituent tag + first span PoS + last span PoS + span length |
|  | s1c\_s1ft\_s1lsw\_s1len | s1: constituent tag + first span PoS + last span word + span length |
|  | s1c\_s1fw\_s1lst\_s1len | s1: constituent tag + first span word + last span PoS + span length |
|  | s1c\_s1fw\_s1lsw\_s1len | s1: constitutent tag + first span word + last span word + span length |

## Our decoding methods for shift-reduce parsing systems.

### Beam search decoder

This is searching algorithm adapted from Zhang Yue’s publication:

* It stores only k-best candidate at each step.
* It also used “early-update” method to train the system.
* Beam size = 16 (standard beam size from Zhang Yue’s publication, in the version of Stanford Parser, they have used beam size = 4)

### A\* search decoder

Exact search is very difficult in feature-based shift-reduce parsing because of many challenge. In order to apply A\* searching algorithm, I have done the following works:

* *Our merging shift-reduce actions*: this approach can guarantee that the number of Shift-Reduce actions will always be 2\*n (n is the length of input sentence).I build a new set of shift-reduce actions including:
  + SHIFT: perform the regular shift action.
  + SHIFT/U-REDUCE(X): perform the regular shift action, then perform unary reduce action with label X.
  + B-REDUCE\_L/R(Y): perform the regular binary reduce action.
  + B-REDUCE\_L/R(Y)/ U-REDUCE(X): perform the regular binary reduce action, then perform unary action with label Y.
* *Building the Dynamic Programming Shift-Reduce Constituent Parser*: I have applied Liang Huang’s method (use for dependency parsing) and CYK algorithm to build a dynamic programming shift-reduce constituent parser. Dynamic Programming can reduce the time complexity of our system down to polynomial. Because of our merging shift-reduce actions, unary nodes and binary nodes with same span will appear in the same step and can be merged together as in Liang Huang’s method. In my knowledge, there is no dynamic programming for shift-reduce constituent parser.
* *Negative score problem*: concerning about this problem, we added an offset to the score of each shift-reduce actions in order to make it positive. Once again, because of our merging shift-reduce actions, the number of shift-reduce actions has always been fixed, therefore adding an offset to each action will not affect the final decoding results.
* *Feature Template:* with the base features, it focus on four top nodes (s0, s1, s2, s3). Following the classic lexical CYK (dynamic programming), the time complexity for each node will be O(n4\*|G|). Therefore, with this base features, we will have the worst-case time complexity about O((n4\*|G|)4) = O(n16\*|G|4) and even greater (because it has also focused on s0l, s0r, s0u, s1l, s1r, s1u). So we propose using the lesser features which reduced the worst-case time complexity down to O((n3\*|G|)2) = O(n6\*|G|2), which is still very high for normal exact search but can be solved by A\* search.

# Evaluation

## Preparation

As we discussed before, I will conduct two phase of experiment:

* First Phase: In [Zhang Yue, 2013], he did not mention about the optimal iteration number (“The optimal iteration number of perceptron learning is determined on the development sets”- that what he said, I suspect that he used some threshold to stop the training). So I will train our parsing system from iteration 1 to 50 to find the corresponding number of iterations.
* Second Phase: with the coressponding iteration number, I will conduct two lines of experiment:
  + Training our system with beam size = 16, 32, 64 with the base features
  + Training our system with beam size = 16, 32, 64 with the lesser features

I used the Stanford PoS tagger to do the PoS tagging job for the input sentences.

Training corpus: section 2 – 21 (WSJ)

Testing corpus: development set (WSJ section 22)

## Experiment results

### First phase experiments

The result of the first phase experiments has been shown as Table 4. The performance of Zhang Yue system on development set is 89.1%. Therefore I select the model at iteration 40th to be the optimal one. All the experiments of the second phase will be conducted with 40 iterations on development set of WSJ.

Table 4. Result of first phase experiment

|  |  |
| --- | --- |
| Iteration | F-score |
| 1 | 81.73 |
| 5 | 85.69 |
| 10 | 87.31 |
| 15 | 88.21 |
| 20 | 88.77 |
| 25 | 88.76 |
| 30 | 88.78 |
| 35 | 89.06 |
| 36 | 89.24 |
| 37 | 88.27 |
| 38 | 88.77 |
| 39 | 88.4 |
| 40 | 89.07 |
| 41 | 88.51 |
| 42 | 88.79 |
| 43 | 89.01 |
| 44 | 89.03 |
| 45 | 89.02 |
| 50 | 89.2 |

### Second phase experiments

Table 5. Result of the second phase experiment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Parameter | F-score on base features | speed on base features | F-score on lesser features | speed on lesser features |
| beam = 16 | 89.07 | 25 sentences/s | 88.72 | 30 sentences/s |
| beam = 32 | 89.87 | 10 sentences/s | 89.33 | 13 sentences/s |
| beam = 64 | 90.3 | 3.4 sentences/s | 90.15 | 4.5 sentences/s |
| A star | N/A | N/A | 90.7 | 0.7 sentences/s,  1.2 sentences/s (len <= 40) |