Experiment report for Shift-Reduce Parsing System

1 Recall the current status of shift-reduce parsing system

1.1 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
1.	s0t_s0c	s0: head PoS + constituent tag
2.	s0w_s0c	s0: head word + constituent tag
3.	s1t_s1c	s1: head PoS + constituent tag
4.	s1w_s1c	s1: head word + constituent tag
5.	s2t_s2c	s2: head PoS + constituent tag
6.	s2w_s2c	s2: head word + constituent tag
7.	s3t_s3c	s3: head PoS + constituent tag
8.	s3w_s3c	s3: head word + constituent tag
9.	N0w_N0t	N0: word + PoS
10.	N1w_N1t	N1: word + PoS
11.	N2w_N2t	N2: word + PoS
12.	N3w_N3t	N3: word + PoS
13.	s0lw_s0lc	s0l: head word + constituent tag
14.	s0rw_s0rc	s0r: head word + constituent tag
15.	s0uw_s0uc	s0u: head word + constituent tag
16.	s1lw_s1lc	s1l: head word + constituent tag
17.	s1rw_s1rc	s1r: head word + constituent tag
18.	s1uw_s1uc	s1u: head word + constituent tag
19.	s0w_s1w	s0: head word + s1: head word
20.	s0w_s1c	s0: head word + s1: constituent tag
21.	s0c_s1w	s0: constituent tag + s1: head word
22.	s0c_s1c	s0: constituent tag + s1: constituent tag
23.	s0w_N0w	s0: head word + N0: word
24.	s0w_N0t	s0: head word + N0: PoS
25.	s0c_N0w	s0: constituent tag + N0: word
26.	s0c_N0t	s0: constituent tag + N0: PoS
27.	N0w_N1w	N0: word + N1: word

28.	N0w_N1t	N0: word + N1: PoS
29.	N0t_N1w	N0: PoS + N1: word
30.	N0t_N1t	N0: PoS + N1: PoS
31.	s1w_N0w	s1: head word + N0: word
32.	s1w_N0t	s1: head word + N0: PoS
33.	s1c_N0w	s1: constituent tag + N0: word
34.	s1c_N0t	s1: constituent tag + N0: PoS
35.	s0c_s1c_s2c	s0: constituent tag + s1: constituent tag + s2: constituent tag
36.	s0w_s1c_s2c	s0: head word + s1: constituent tag + s2: constituent tag
37.	s0c_s1w_s2c	s0: constituent tag + s1: head word + s2: constituent tag
38.	s0c_s1c_s2w	s0: constituent tag + s1: constituent tag + s2: head word
39.	s0c_s1c_N0t	s0: constituent tag + s1: constituent tag + N0: PoS
40.	s0w_s1c_N0t	s0: head word + s1: constituent tag + N0: PoS
41.	s0c_s1w_N0t	s0: constituent tag + s1: head word + N0: PoS
42.	s0c_s1c_N0w	s0: constituent tag + s1: constituent tag + N0: word

Table 3. List of lesser features

Id	Feature	Description
1.	N0w_N0t	N0: word + PoS
2.	N1w_N1t	N1: word + PoS
3.	N2w_N2t	N2: word + PoS
4.	N3w_N3t	N3: word + PoS
5.	s0c_s0ft	s0: constituent tag + first span PoS
6.	s0c_s0fw	s0: constituent tag + first span word
7.	s0c_s0lst	s0: constituent tag + last span PoS
8.	s0c_s0lsw	s0: constituent tag + last span word
9.	s0c_s0at	s0: constituent tag + after span PoS
10.	s0c_s0aw	s0: constituent tag + after span word

11.	s0c_s0bt	s0: constituent tag + before span PoS
12.	s0c_s0bw	s0: constituent tag + before span word
13.	s0c_s0len	s0: constituent tag + span length
14.	s1c_s1ft	s1: constituent tag + first span PoS
15.	s1c_s1fw	s1: constituent tag + first span word
16.	s1c_s1lst	s1: constituent tag + last span PoS
17.	s1c_s1lsw	s1: constituent tag + last span word
18.	s1c_s1at	s1: constituent tag + after span PoS
19.	s1c_s1aw	s1: constituent tag + after span word
20.	s1c_s1bt	s1: constituent tag + before span PoS
21.	s1c_s1bw	s1: constituent tag + before span word
22.	s1c_s1len	s1: constituent tag + span length
23.	s0c_s0Shape	s0: constituent tag + span shape
24.	s1c_s1Shape	s1: constituent tag + span shape
25.	s0c_s0ft_s0lst	s0: constituent tag + first span PoS + last span word
26.	s0c_s0fw_s0lsw	s0: constituent tag + first span word + last span word
27.	s1c_s1ft_s1lst	s1: constituent tag + first span PoS + last span PoS
28.	s1c_s1fw_s1lsw	s1: constituent tag + first span word + last span word
29.	s0c_s0ft_s0len	s0: constituent tag + first span PoS + span length
30.	s0c_s0fw_s0len	s0: constituent tag + first span word + span length
31.	s0c_s0lst_s0len	s0: constituent tag + last span PoS + span length
32.	s0c_s0lsw_s0len	s0: constituent tag + last span word + span length
33.	s1c_s1ft_s1len	s1: constituent tag + first span PoS + span length
34.	s1c_s1fw_s1len	s1: constituent tag + first span word + span length
35.	s1c_s1lst_s1len	s1: constituent tag + last span PoS + span length
36.	s1c_s1lsw_s1len	s1: constituent tag + last span word + span length
37.	s0c_s0ft_s0lst_s0len	s0: constituent tag + first span PoS + last span PoS + span length
38.	s0c_s0ft_s0lsw_s0len	s0: constituent tag + first span PoS + last span word + span length
39.	s0c_s0fw_s0lst_s0len	s0: constituent tag + first span word + last span PoS + span length

40.	s0c_s0fw_s0lsw_s0len	s0: constitutent tag + first span word + last span word + span length
41.	s1c_s1ft_s1lst_s1len	s1: constituent tag + first span PoS + last span PoS + span length
42.	s1c_s1ft_s1lsw_s1len	s1: constituent tag + first span PoS + last span word + span length
43.	s1c_s1fw_s1lst_s1len	s1: constituent tag + first span word + last span PoS + span length
44.	s1c_s1fw_s1lsw_s1len	s1: constitutent tag + first span word + last span word + span length

1.2 Our decoding methods for shift-reduce parsing systems.

1.2.1 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)

1.2.2 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 $O(n^4*|G|)$. Therefore, with this base features, we will have the worst-case time complexity about $O((n^4*|G|)^4) = O(n^{16}*|G|^4)$ 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 $O((n^3*|G|)^2) = O(n^6*|G|^2)$, which is still very high for normal exact search but can be solved by A* search.

2 Evaluation

2.1 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)

2.2 Experiment results

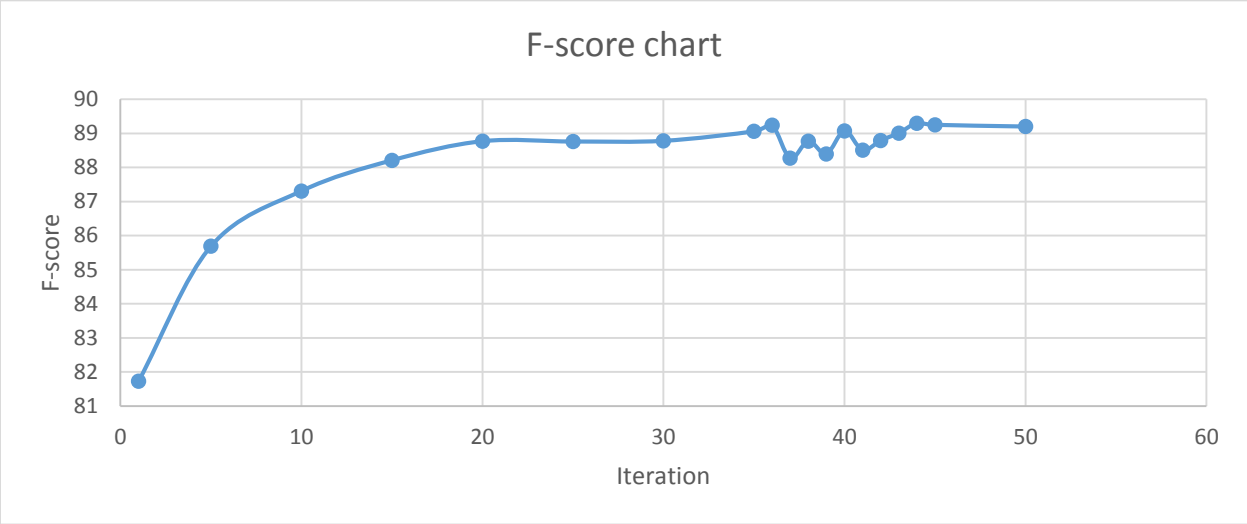
2.2.1 First phase experiments

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



2.2.2 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)