

## Appendix A Preliminary Experiments

### A.A Skip Connection

Skip connections of composition model help address the vanishing gradient problem, and following experiments show that they are necessary to integrate pre-trained BERT (Devlin et al., 2018) parameters with the model:

Model	UAS	LAS
BERT StateTr	94.78	92.06
BERT StateTr without skip	93.16	90.51

Table 1: Preliminary experiments on the development set of WSJ Penn Treebank for BERT StateTr model with/without skip connections.

### A.B Position Embeddings

Following experiments show that using position embeddings for the whole sequence achieves better performance than applying separate position embeddings for each segment:

Model	UAS	LAS
BERT StateTr	94.78	92.06
BERT StateTr with separate pos	93.10	90.39

Table 2: Preliminary experiments on the development set of WSJ Penn Treebank for BERT StateTr model, and its variation with separate position embeddings for each section.

## Appendix B Composition Model

Previous work has shown that recursive neural networks are capable of inducing a representation for complex phrases by recursively embedding sub-phrases (Socher et al., 2011, 2014, 2013; Hermann and Blunsom, 2013). Dyer et al. (2015) showed that this is an effective technique for embedding the partial parse subtrees specified by the parse history in transition-based dependency parsing. Since a word in a dependency tree can have a variable number of dependents, they combined the dependency relations incrementally as they are specified by the parser.

We extend this idea by using a feed-forward neural network with  $\text{Tanh}$  as the activation function and skip connections. For every token in position  $i$  on the stack or buffer, after deciding on step  $t$ , the composition model computes a vector  $C_{a,i}^{t+1}$  which is added to the input embedding for that token:

$$C_{a,i}^{t+1} = \text{Comp}((\psi_{a,i}^t, \omega_{a,i}^t, l_{a,i}^t)) + \psi_{a,i}^t \quad (11)$$

where:  $a \in \{S, B\}$

where the  $\text{Comp}$  function is a one-layer feed forward neural network, and  $(\psi_{a,i}^t, \omega_{a,i}^t, l_{a,i}^t)$  represents the most recent dependency relation with head  $\psi_{a,i}^t$  specified by the decision at step  $t$  for element in position  $i$  in the stack or buffer. In arc-standard parsing, the only word which might have received a new dependent by the previous decision is the word on the top of the stack,  $i=1$ . This gives us the following definition

of  $(\psi_{a,i}^t, \omega_{a,i}^t, l_{a,i}^t)$ :

$$\left\{ \begin{array}{l} \text{RIGHT-ARC}(l): \\ \psi_{S,1}^t = C_{S,2}^t, \omega_{S,1}^t = C_{S,1}^t, l_{S,1}^t = l, \\ \psi_{S,i \neq 1}^t = C_{S,i+1}^t, \omega_{S,i \neq 1}^t = \omega_{S,i+1}^t, l_{S,i \neq 1}^t = l_{S,i+1}^t \\ \psi_{B,i}^t = C_{B,i}^t \\ \text{LEFT-ARC}(l): \\ \psi_{S,1}^t = C_{S,1}^t, \omega_{S,1}^t = C_{S,2}^t, l_{S,1}^t = l, \\ \psi_{S,i \neq 1}^t = C_{S,i+1}^t, \omega_{S,i \neq 1}^t = \omega_{S,i+1}^t, l_{S,i \neq 1}^t = l_{S,i+1}^t \\ \psi_{B,i}^t = C_{B,i}^t \\ \text{SHIFT}: \\ \psi_{S,1}^t = C_{B,1}^t, \omega_{S,1}^t = \omega_{B,1}^t, l_{S,1}^t = l_{B,1}^t \\ \psi_{S,i \neq 1}^t = C_{S,i-1}^t, \omega_{S,i \neq 1}^t = \omega_{S,i-1}^t, l_{S,i \neq 1}^t = l_{S,i-1}^t \\ \psi_{B,i}^t = C_{B,i+1}^t, \omega_{B,i}^t = \omega_{B,i+1}^t, l_{B,i}^t = l_{B,i+1}^t \\ \text{SWAP}: \\ \psi_{S,1}^t = C_{S,1}^t \\ \psi_{S,i \neq 1}^t = C_{S,i+1}^t, \omega_{S,i \neq 1}^t = \omega_{S,i+1}^t, l_{S,i \neq 1}^t = l_{S,i+1}^t \\ \psi_{B,1}^t = C_{S,2}^t, \omega_{B,1}^t = \omega_{S,2}^t, l_{B,1}^t = l_{S,2}^t \\ \psi_{B,i \neq 1}^t = C_{B,i-1}^t, \omega_{B,i \neq 1}^t = \omega_{B,i-1}^t, l_{B,i \neq 1}^t = l_{B,i-1}^t \end{array} \right. \quad (12)$$

where  $C_{S,1}^t$  and  $C_{S,2}^t$  are the embeddings of the top two elements of the stack at time step  $t$ , and  $C_{B,1}^t$  is the embedding of the word on the front of the buffer at time  $t$ .  $l_{a,i}^t \in R^m$  is the label embedding of the specified relation, including its direction. For all words which have not received a new dependent, the composition is computed anyway with the most recent dependent and label (with a [NULL] dependent and label of that position [L-NULL] if there is no dependency relation with element  $i$  as the head).<sup>13</sup>

At  $t=0$ ,  $C_{a,i}^t$  is set to the initial token embedding  $T_{w_i}$ . The model then computes Equation 11 iteratively at each step  $t$  for each token on the stack or buffer.

There is a skip connection in Equation 11 to address the vanishing gradient problem. Also, preliminary experiments showed that without this skip connection to bias the composition model towards the initial token embeddings  $T_{w_i}$ , integrating pre-trained BERT (Devlin et al., 2018) parameters into the model did not work.

<sup>13</sup>Preliminary experiments indicated that not updating the composition embedding for these cases resulted in worse performance.

## Appendix C Example of the Graph-to-Graph Transformer parsing

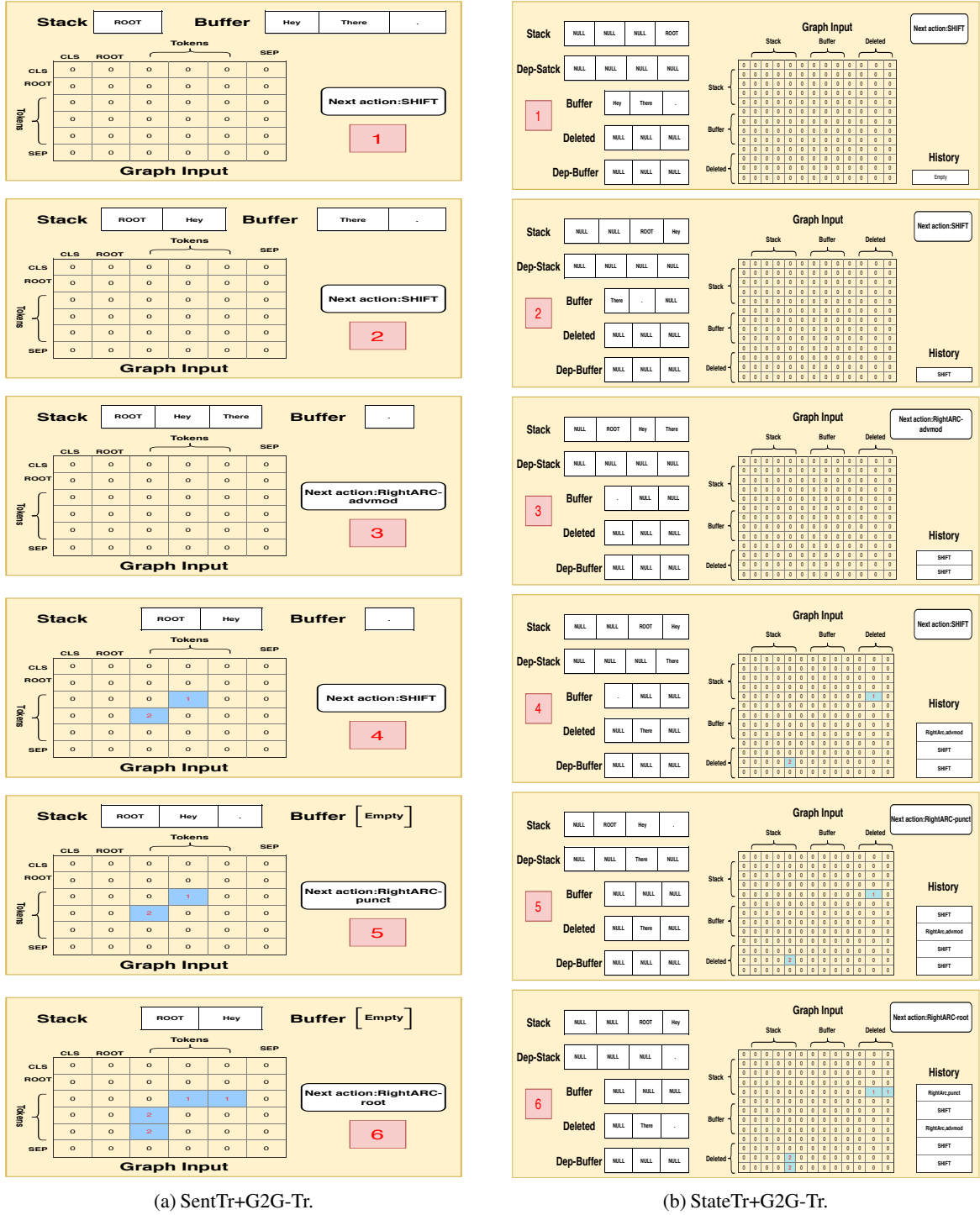


Figure 4: Example of Graph-to-Graph Transformer model integrated with SentTr and StateTr on UD English Treebank (initial sentence: "Hey There .").

## Appendix D Description of Treebanks

### D.A English Penn Treebank Description

The dataset can be found [here](#) under LDC licence. Stanford PoS tagger and constituency converter can be downloaded from [here](#) and [here](#), respectively. Here is the detailed information of English Penn Treebank:

Language	Version	Non-projectivity ratio	Train size(2-21)	Development size(22,24)	Test size(23)
English	3	0.1%	39'832	3'046	2'416

Table 3: Description of English Penn Treebank.

## D.B UD Treebanks Description

UD Treebanks v2.3 are provided in [here](#). Pre-processing tools can be found [here](#).

Language	Family	Treebank	Order	Train size	Development size	Test size	Non-projectivity ratio
Arabic	non-IE	PADT	VSO	6.1K	0.9K	0.68K	9.2%
Basque	non-IE	BDT	SOV	5.4K	1.8K	1.8K	33.5%
Chinese	non-IE	GSD	SVO	4K	0.5K	0.5K	0.6%
English	IE	EWT	SVO	12.5K	2k	2.1K	5.3%
Finnish	non-IE	TDT	SVO	12.2K	1.3K	1.5K	6.2%
Hebrew	non-IE	HTB	SVO	5.2K	0.48K	0.49K	7.6%
Hindi	IE	HDTB	SOV	13.3K	1.7K	1.7K	13.8%
Italian	IE	ISDT	SVO	13.1K	0.56K	0.48K	1.9%
Japanese	non-IE	GSD	SOV	7.1K	0.51K	0.55K	2.7%
Korean	non-IE	GSD	SOV	4.4K	0.95K	0.99K	16.2%
Russian	IE	SynTagRus	SVO	48.8K	6.5K	6.5K	8.0%
Swedish	IE	Talbanken	SVO	4.3K	0.5K	1.2K	3.3%
Turkish	non-IE	IMST	SOV	3.7K	0.97K	0.97K	11.1%

Table 4: Description of languages chosen from UD Treebanks v2.3.

## Appendix E Running Details of Proposed Models

We provide the number of parameters and average run times for each model. For a better understanding, average run time is computed per transition (Second/transition). All experiments are computed with graphics processing unit (GPU), specifically the NVIDIA V100 model. The total number of transitions in the train and development sets are 79664 and 6092, respectively.

Model	No. parameters	Train (sec/transition)	Evaluation (sec/transition)	Evaluation (sent/sec)	Evaluation (token/sec)
StateTr	90.33M	0.098	0.031	16.2	388.13
StateTr+G2GTr	105.78M	0.226	0.071	7.05	168.91
SentTr	63.23M	0.112	0.026	19.27	460.6
SentTr+G2GTr	63.27M	0.138	0.031	16.2	388.13

Table 5: Running details of our models on WSJ Penn Treebank.

## Appendix F Hyper-parameters for our parsing models

For hyper-parameter selection, we use manual tuning to find the best numbers. For BERT (Devlin et al., 2018) hyper-parameters, we apply the same optimization strategy as suggested by Wolf et al. (2019). For classifiers and composition model, we use a one-layer feed-forward neural network for simplicity. Then, we pick hyper-parameters based on previous works (Devlin et al., 2018; Dyer et al., 2015). We use two separate optimisers for pre-trained parameters (BERT here) and randomly initialised parameters for better convergence that is shown to be useful in Kondratyuk and Straka (2019). Early stopping (based on LAS) is used during training. The only tuning strategy that has been tried is to use one optimiser for all parameters or two different optimisers for pre-trained parameters and randomly initialised ones. For the latter case, Learning rate for randomly initialised parameters is set to  $1e-4$ . Results of different variations on the development set of WSJ Penn Treebank are as follows:

Model	UAS	LAS
BERT StateTr with one optimiser	94.66	91.94
BERT StateTr with two optimisers	94.24	91.67
Expected (Average) Performance	94.45	91.81
BERT StateTr+G2GTr with one optimiser	94.96	92.88
BERT StateTr+G2GTr with two optimisers	94.75	92.49
Expected (Average) Performance	94.86	92.69
BERT SentTr with one optimiser	95.34	93.29
BERT SentTr with two optimisers	95.49	93.29
Expected (Average) Performance	95.42	93.29
BERT SentTr+G2GTr with one optimiser	95.27	93.18
BERT SentTr+G2GTr with two optimisers	95.66	93.60
Expected (Average) Performance	95.47	93.40

Table 6: Results on the development set of WSJ Penn Treebank for different optimisation strategy.

Hyper-parameters for training our models are defined as <sup>14</sup>:

Component	Specification
<b>Optimiser</b>	BertAdam
Learning Rate	1e-5
Base Learning Rate	1e-4
Adam Betas( $b_1, b_2$ )	(0.9, 0.999)
Adam Epsilon	1e-6
Weight Decay	0.01
Max-Grad-Norm	1
Warm-up	0.01
<b>Self-Attention</b>	
No. Layers( $n$ )	6
No. Heads	12
Embedding size	768
Max Position Embedding	512
BERT model	bert-base-uncased
<b>Classifiers</b>	
No. Layers	2
Hidden size(Exist)	500
Hidden size(Relation)	100
Drop-out	0.05
Activation	<i>ReLU</i>
<b>History Model</b>	LSTM
No. Layers	2
Hidden Size	768
<b>Composition Model</b>	
No. Layers	2
Hidden size	768
Epochs	12

Table 7: Hyper-parameters for training our models.

<sup>14</sup>For UD Treebanks, we train our model for 20 epochs, and use "bert-multilingual-cased" for the initialisation. We use pre-trained BERT models of <https://github.com/google-research/bert>.

## Appendix G Pseudo-Code of Graph Input Mechanism

**Algorithm 1** Pseudo-code of building graph input matrix for StateTr+G2GTr model.

---

```

1: Graph Sentence(input of attention):  $P$ 
2: Graph Input:  $G$ 
3: Actions:  $A = (a_1, \dots, a_T)$ 
4: Input:  $(S, B, D)$ 
5: for  $k \leftarrow 1, T$  do
6:   if  $a_k = \text{SHIFT or SWAP}$  then
7:     continue
8:   else
9:     new relation:  $i \xrightarrow{l} j$ 
10:     $G_{i,j} = 1$ 
11:     $G_{j,i} = 2$ 
12:    pop  $x_j$  from stack
13:    change mask of  $x_j$  to one
14:    add  $l$  to input embedding of  $x_j$ 
15:     $P$ :select  $G$  based on Input  $(S, B, D)$ 
16:   end if
17: end for

```

---

**Algorithm 2** Pseudo-code of building graph input matrix for SentTr+G2GTr model.

---

```

1: Graph Sentence(input of attention):  $P$ 
2: Graph Input:  $G$ 
3: Actions:  $A = (a_1, \dots, a_T)$ 
4: Input: initial tokens
5: for  $k \leftarrow 1, T$  do
6:   if  $a_k = \text{SHIFT or SWAP}$  then
7:     continue
8:   else
9:     new relation:  $i \xrightarrow{l} j$ 
10:     $G_{i,j} = 1$ 
11:     $G_{j,i} = 2$ 
12:    add  $l$  to input embedding of  $j$ -th word
13:     $P = G$ 
14:   end if
15: end for

```

---

## Appendix H UD Treebanks Results

BERT SentTr+G2GTr results:

Language	Test set-UAS	Dev set-UAS	Dev set-LAS
Arabic	87.65	87.01	82.64
Basque	87.17	86.53	83.25
Chinese	89.74	88.36	85.79
English	92.05	93.05	91.3
Finnish	91.46	91.37	89.72
Hebrew	90.85	91.92	89.55
Hindi	95.77	95.86	93.17
Italian	95.15	95.22	93.9
Japanese	96.21	96.68	96.04
Korean	89.42	87.57	84.94
Russian	94.42	94.0	92.70
Swedish	92.49	87.26	85.39
Turkish	74.23	72.52	66.05
Average	90.51	89.80	87.27

Table 8: Dependency scores of BERT SentTr+G2GTr model on the development and test sets of UD Treebanks.

BERT StateTr+G2GTr results:

Language	Test set-UAS	Dev set-UAS	Dev set-LAS
Arabic	86.85	86.41	81.73
Basque	80.91	80.01	73.2
Chinese	87.90	86.64	84.15
English	90.91	91.85	90.11
Finnish	84.35	82.91	78.73
Hebrew	89.51	90.36	87.85
Hindi	95.65	95.92	93.30
Italian	93.5	93.61	92.18
Japanese	95.99	96.18	95.58
Korean	84.35	82.13	77.78
Russian	93.87	93.41	92.09
Swedish	92.49	90.72	88.36
Turkish	65.99	65.92	56.96
Average	87.87	87.39	84.01

Table 9: Dependency scores of BERT StateTr+G2GTr model on the development and test sets of UD Treebanks.

## Appendix I Error-Analysis

### I.A Dependency Length

Model	ROOT	1	2	3	4	5	6	7	8	9	$\geq 10$
BERT StateTr	96.40	94.30	93.15	91.00	87.80	85.19	82.80	81.25	80.49	82.54	84.82
BERT StateCLSTr	95.80	93.75	92.25	89.60	86.45	82.77	80.58	79.62	79.98	78.90	81.74
BERT StateTr+G2GTr	97.10	94.45	93.60	92.20	89.80	87.54	86.00	84.60	84.45	85.55	86.86
BERT StateTr+G2CLSTr	96.50	94.10	93.00	91.45	88.65	85.74	84.25	82.55	81.80	82.25	85.50
BERT StateTr+G2GTr+C	96.95	94.25	93.25	91.30	88.30	85.74	83.50	82.75	83.10	83.95	86.15

Table 10: labelled F-Score vs dependency relation length

Model	ROOT	1	2	3	4	5	6	7	8	9	$\geq 10$
BERT StateTr	3046	31245	14478	7565	4153	2461	1681	1185	933	808	5415
BERT StateCLSTr	3046	31409	14473	7553	4131	2406	1641	1153	923	832	5403
BERT StateTr+G2GTr	3047	31240	14457	7572	4171	2465	1688	1188	941	811	5390
BERT StateTr+G2CLSTr	3046	31249	14447	7537	4143	2453	1701	1193	953	814	5434
BERT StateTr+G2GTr+C	3047	31304	14430	7514	4137	2449	1693	1182	951	830	5433
Gold bins	3046	31126	14490	7551	4155	2508	1698	1195	953	821	5427

Table 11: Size of each bin based on dependency length

### I.B Distance to Root

Model	ROOT	1	2	3	4	5	6	7	8	9	$\geq 10$
BERT StateTr	96.40	91.35	91.00	91.45	91.80	91.50	92.10	91.75	91.90	90.06	93.06
BERT StateCLSTr	95.80	90.55	90.20	90.25	90.70	90.65	91.10	89.95	89.20	88.24	87.69
BERT StateTr+G2GTr	97.10	92.70	92.10	92.40	92.05	92.05	92.55	91.99	92.39	90.49	94.54
BERT StateTr+G2CLSTr	96.50	91.60	91.30	91.55	91.65	91.55	92.10	91.90	91.10	89.93	93.48
BERT StateTr+G2GTr+C	96.95	92.10	91.45	91.55	91.70	91.45	92.35	91.75	92.59	89.50	91.77

Table 12: labelled F-Score vs distance to root

Model	ROOT	1	2	3	4	5	6	7	8	9	$\geq 10$
BERT StateTr	3046	16081	15965	12999	9301	6488	3964	2242	1343	740	801
BERT StateCLSTr	3046	15963	15798	12793	9222	6419	3974	2331	1358	827	1239
BERT StateTr+G2GTr	3047	16024	15889	12875	9415	6463	3995	2349	1347	743	823
BERT StateTr+G2CLSTr	3046	16064	15975	12971	9327	6488	3963	2279	1316	708	833
BERT StateTr+G2GTr+C	3047	16079	15947	13020	9419	6481	3930	2259	1274	701	813
Gold bins	3046	16002	16142	13064	9403	6411	3923	2298	1298	702	681

Table 13: Size of each bin based on distance to root

### I.C Sentence Length

Model	1-9	10-19	20-29	30-39	40-49	$\geq 50$
BERT StateTr	96.1	95.6	95.0	94.4	93.9	90.2
BERT StateCLSTr	94.6	95.0	94.4	93.9	93.0	80.2
BERT StateTr+G2GTr	95.1	95.9	95.3	94.6	94.4	91.2
BERT StateTr+G2CLSTr	94.7	95.1	94.8	94.2	93.2	89.4
BERT StateTr+G2GTr+C	94.7	95.3	95.1	94.5	93.4	86.7

Table 14: LAS vs. sentence length

## I.D Dependency Type Analysis

Type	StateTr+G2GTr	StateTr	StateTr+G2CLSTr
acomp	68.62	58.60 (-31.9%)	66.04 (-8.2%)
advcl	82.75	70.68 (-70.0%)	81.85 (-5.2%)
advmod	83.85	84.40 (3.4%)	84.35 (3.1%)
amod	92.55	92.25 (-4.1%)	92.30 (-3.4%)
appos	87.85	84.94 (-23.9%)	83.25 (-37.8%)
aux	98.55	98.35 (-13.8%)	98.45 (-6.9%)
auxpass	96.65	96.04 (-18.0%)	95.84 (-24.0%)
cc	90.90	90.45 (-4.9%)	88.80 (-23.1%)
ccomp	89.49	81.82 (-73.0%)	87.56 (-18.4%)
conj	86.45	84.70 (-12.9%)	84.15 (-17.0%)
cop	93.08	92.62 (-6.5%)	91.58 (-21.7%)
csubj	76.94	67.93 (-39.0%)	70.83 (-26.5%)
dep	54.66	50.88 (-8.3%)	51.99 (-5.9%)
det	98.25	98.30 (2.9%)	98.00 (-14.3%)
discourse	15.40	15.40 (-0.0%)	28.60 (15.6%)
dobj	94.85	94.10 (-14.6%)	93.95 (-17.5%)
expl	96.39	94.99 (-38.8%)	96.39 (-0.0%)
infmod	87.38	79.19 (-64.9%)	84.93 (-19.4%)
iobj	88.01	90.66 (22.1%)	84.24 (-31.4%)
mark	95.04	95.19 (2.9%)	94.84 (-4.1%)
mwe	86.46	89.71 (24.0%)	88.36 (14.1%)
neg	95.75	94.84 (-21.4%)	93.78 (-46.2%)
nn	94.25	94.10 (-2.6%)	93.65 (-10.5%)
npadvmod	91.89	92.75 (10.5%)	90.64 (-15.5%)
nsubj	96.35	95.55 (-21.9%)	95.60 (-20.6%)
nsubjpass	95.49	92.70 (-61.9%)	94.08 (-31.1%)
num	95.25	94.89 (-7.4%)	95.15 (-2.0%)
number	92.50	90.65 (-24.6%)	92.10 (-5.3%)
parataxis	69.59	62.89 (-22.1%)	72.10 (8.2%)
partmod	82.11	72.82 (-52.0%)	79.74 (-13.3%)
pcomp	88.12	86.49 (-13.7%)	85.77 (-19.8%)
pobj	97.15	96.95 (-7.0%)	96.60 (-19.3%)
poss	97.60	97.15 (-18.7%)	97.70 (4.2%)
possessive	98.29	98.04 (-14.5%)	98.44 (8.9%)
preconj	85.71	84.65 (-7.5%)	84.65 (-7.5%)
predet	79.34	79.34 (-0.0%)	77.44 (-9.2%)
prep	90.25	89.90 (-3.6%)	89.50 (-7.7%)
prt	84.48	83.42 (-6.9%)	83.10 (-8.9%)
punct	88.45	88.05 (-3.5%)	87.65 (-6.9%)
quantmod	86.97	84.40 (-19.7%)	84.49 (-19.0%)
rcmod	86.84	76.38 (-79.5%)	83.91 (-22.3%)
root	97.15	96.40 (-26.3%)	96.50 (-22.8%)
tmod	86.02	86.38 (2.6%)	85.03 (-7.1%)
xcomp	90.75	84.44 (-68.2%)	89.75 (-10.8%)

Table 15: F-score of StateTr, StateTr+G2GTr, and StateTr+G2CLSTr models for the dependency types on the development set of WSJ Treebank. Relative error reduction is computed by considering StateTr+G2GTr scores as the reference.