A Appendices

A.1 Related Work

Traditional coreference resolution studies are divided into rule- and machine learning-based methods. In the rule-based method, Stanford's model (Lee et al., 2013), applied to multi-pass sieve using pronouns, entity attributes, named entity information, and so on. In the statistics-based, various coreference models have been proposed such as mention-pair (Ng and Cardie, 2002; Ng, 2010), mention-ranking (Wiseman et al., 2015; Clark and Manning, 2016a) and entity-level models (Haghighi and Klein, 2010; Clark and Manning, 2016b).

Lee et al. (2017) defined mentions as span representations and proposed a span ranking model based on long short-term memory (LSTM, (Hochreiter and Schmidhuber, 1997)) for all spans in the document. As span representations could reflect the contextual information from LSTM, but the other two spans are interpreted as a related entity. This phenomenon results in local consistency errors that yield erroneous coreference resolutions. Hence, Lee et al. (2018) performed the attention mechanism to resolve coreference using a high-order function. The end-to-end model of (Lee et al., 2017, 2018) showed the superior performance in English coreference resolution however, the complexity of $O(n^4)$ is considering all spans and span pairs of the document. Zhang et al. (2018) is based on the (Lee et al., 2017), which replaced the concat attention score into the biaffine attention score to calculate the conference score. Also, it performed the multi-task learning process that also calculates the loss for the mention score.

Simple recurrent units (SRU) (Lei et al., 2017) architecture solves the vanishing gradient problem that occurs when back-propagation of the recurrent neural network (RNN). SRU, which is one of RNN types such as gated recurrent unit architecture (GRU) (Cho et al., 2014) and LSTM, is less computational complexity than other RNN types because the SRU encodes hidden states using a feed-forward neural gate and recurrent cell in a layer.

Recently, a variety of downstream studies using BERT (Bidirectional Encoder Representations from Transformer, Vaswani et al. (2017); Devlin et al. (2019)) which have been pre-trained with large amounts of data, have been conducted in natural language processing tasks (Joshi et al., 2019b; Zhang et al., 2019; Park et al., 2019a; Wang et al., 2019). A BERT-coref study was also conducted in the English coreference resolution task, and a more effective SpanBERT (Joshi et al., 2019a) for coreference resolution has also been studied, with dramatic gains in GAP (Webster et al., 2018) and OntoNotes (Pradhan et al., 2012) datasets. A qualitative assessment of BERT-coref showed that BERT is significantly better at distinguishing unique entities and concepts.

A.2 Data Format for Our Model

The following example shows input sequence, head list and decoder output format.

- **Input sequence for BERT**: "[CLS] 그리스/NNP_로 마/NNG_신화/NNG_에서/JKB_바 카스/NNP_이/VCP_라고/EC_도/JX_불리/VV_는/ETM_술/NNG_의/JKG_신/NNG_[SEP]"
- **Heads**: "그리스/NNP, 로마/NNG, 신화/NNG, 바카스/NNP, 술/NNG, 신/NNG"
- **Heads applied by BPE**: "그리스/NNP_, 로마/NNG_, 신화/NNG_, 바, 술/NNG_, 신/NNG_"
- **Head list**: [0, 1, 2, 3, 5, 12, 14]
- **Decoder output**: [0, 0, 0, 0, 0, 5, 5]

We add [CLS] and [SEP] to match the input sequence to the BERT format. The Heads is an example of heads included in a sentence, and the Heads applied by BPE is an example of heads with BPE applied. BPE divides words into subwords. The head divided into subwords uses the first token as the representative of the head. In the example of the Heads applied by BPE, the representative of the BPE-applied head '바' (Ba) and '카스/NNP' (cchus/NNP) is '바' (Ba). The Head list is the position of the head in the sentence that matches the BERT input format, which is input to the decoder. The head list is a target class. The decoder output is a position where the coreference resolves in the head list. Since '바' (Ba) is first mention in the entity of Bacchus, '바' (Ba) outputs its own location of 5. '신/NNG' (a god) outputs position 5 because it is linked to '바' (Ba). We then change the output to word units via post-processing.

A.3 Overall Performance

Please refer to Table 6 for full performance on all metrics, and dev set results for Table 7.

Model	MUC			B^3			$CEAF_{\phi_4}$			CoNLL
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Avg. F1
e2e-coref (Lee et al., 2017)	66.9	55.2	60.5	64.5	53.1	58.2	66.1	53.9	59.4	59.4
c2f-coref (Lee et al., 2018)	68.3	56.4	61.8	59.0	53.4	59.0	66.4	54.4	59.8	60.2
BERT-coref (Joshi et al., 2019b)	71.7	65.0	68.2	69.3	63.0	66.0	72.2	62.4	66.9	67.0
BERT-SRU enc-dec (Google)	67.7	61.9	64.6	65.7	59.8	62.6	68.5	58.8	63.3	63.5
BERT-SRU enc-dec (single)	67.3	67.3	67.3	64.8	65.3	65.1	69.5	**63.5**	66.3	66.2
BERT-SRU enc-dec (ensemble)	**72.3**	67.6	**69.9**	**70.0**	65.2	**67.5**	**75.0**	63.0	**68.5**	**68.6**
BERT-SRU enc-dec (KD)	68.0	**68.2**	68.1	65.6	**66.0**	65.8	71.1	62.9	66.7	66.9

Table 6: Experimental results on the test set of the Korean data from ETRI wiseQA. The final column (CoNLL Avg. F1) is the main evaluation metric, averaged by the F1 of MUC, B^3, and $CEAF_{\phi_4}$. Based on the Korean head-final, the coreference resolution score is calculated based on the head of the mentions.

Model	**Avg. F1**	Δ
BERT-SRU ptr-net (single)	70.83	-
− fine-tuning	64.74	−6.09
BERT-SRU ptr-net (Google)	67.38	−3.45
`BERT-coref`	68.62	−2.21

Table 7: Dev set results. We evaluate the performance of models using different BERT.

A.4 Optimizing Hyperparameters

We perform hyperparameter optimization on the baseline model of the BERT-SRU Pointer Networks, which is not applied to the head target class component. We optimize hyperparameters for the development set, and hyperparameter optimization proceeds for the feature embedding size, the number of RNN hidden layer dimensions, and the number of biaffine hidden layer dimensions. We set the number of dimensions to 50, 100, 200, 400, 800, 1600, respectively, to find the hyperparameters that give the best performance.

Optimizing Dimension Size of Feature Embedding In Table 8, we perform an optimization of the feature embedding size and our model shows the best performance when the embedding size is 1600. At this time, we could see that the overall performance improves in proportion to the size of the embedding dimension according to Table 8.

Optimizing Size of RNN Hidden States The optimization of the number of RNN hidden layer dimensions is as shown in Table 9, and when the hidden state size is 800, the performance is as good as Table 8. We consider that our model with the number of moderately large dimensions shows good performance because the hidden state **e** of equation 1 is that the hidden state of the BERT and the hidden state of the feature are concatenated.

Optimizing Size of Biaffine Hidden States Table 10 shows the optimization of the number of biaffine hidden layer dimensions, and when the number of hidden layer dimensions is 50, the performance 69.72% of CoNLL F1 is shown as in the previous tables. We perform modeling by applying the head target class component based on the optimized hyperparameters. As a result, the performance of the single model shows 70.83% of CoNLL F1.

A.5 Optimizing RNN types

Table 11 compares performance by RNN types such as SRU, LSTM, and GRU. We choose the RNN type suitable for Korean coreference resolution and optimize the number of layers of each RNN type. The optimal RNN type and the number of layers are 70.83% F1 with 2-layers SRU. Because the SRU uses a highway network ((Srivastava et al., 2015)), a skip connection is used to allow the gradient to directly propagate to the previous layer; the information loss is small even if the stack is deepened.

A.6 Ensemble Knowledge Distillation

Ensemble Table 12 shows the performances of ten single models with different random seed and ensemble models on the dev set. We are interested in how the proposed model performs under different random initial conditions. Our model observes consistent performance regardless of 10 different initializations. The lowest performance among the 10-models is 70.04% F1, and the mean F1 score is 70.37%, both of which still outperforms the 68.62 F1 of the Korean BERT-coref from Joshi et al. (2019b). We perform a maximum score ensemble and an average score of the ensemble for 10-models. The maximum score ensemble is 72.26% F1, and the average score of the ensemble is 72.23% F1.

	MUC	B^3	$CEAF_{\phi_4}$		CoNLL	
Number of dimensions	F1	F1	F1	Pre.	Rec.	Avg. F1
50	69.71	66.85	64.60	63.05	71.65	67.05
100	69.22	66.86	65.76	62.66	72.65	67.28
200	70.14	67.75	65.85	65.80	70.21	67.91
400	70.42	67.93	66.40	65.38	71.42	68.25
800	70.56	67.82	66.32	65.11	71.69	68.23
1600	**71.92**	**69.16**	**68.08**	**66.85**	**72.85**	**69.72**

Table 8: Optimizing number of feature embedding size on the Korean ETRI dev set.

	MUC	B^3	$CEAF_{\phi_4}$		CoNLL	
Number of dimensions	F1	F1	F1	Pre.	Rec.	Avg. F1
50	69.76	67.68	**69.24**	66.48	71.54	68.89
100	69.72	66.71	65.16	64.41	70.25	67.20
200	69.97	67.14	65.44	64.27	71.14	67.52
400	69.26	67.52	68.63	**66.88**	70.17	68.47
800	**71.92**	**69.16**	68.08	66.85	**72.85**	**69.72**
1600	70.64	68.32	69.48	67.22	72.00	69.48

Table 9: Optimizing number of RNN hidden layer dimensions on the Korean ETRI dev set.

	MUC	B^3	$CEAF_{\phi_4}$		CoNLL	
Number of dimensions	F1	F1	F1	Pre.	Rec.	Avg. F1
50	**71.92**	**69.16**	**68.08**	66.85	72.85	**69.72**
100	70.94	68.36	66.77	67.07	70.48	68.69
200	70.77	68.22	66.24	64.44	72.93	68.41
400	70.97	68.13	66.56	63.92	**73.92**	68.55
800	70.30	67.39	65.02	63.43	72.36	67.57
1600	70.81	68.23	66.21	**67.22**	69.74	68.42

Table 10: Optimizing number of Biaffine hidden layer dimensions on the Korean ETRI dev set.

RNN type	#Layer	Avg. F1
SRU	1	69.30
SRU	2	**70.83**
GRU	1	69.77
GRU	2	69.74
LSTM	1	69.31
LSTM	2	68.55

Table 11: Optimizing RNN type and the number of layers on the Korean ETRI dev set.

Seed#	Avg. F1	Seed#	Avg. F1
Seed 1	70.04	Seed 6	70.23
Seed 2	70.31	Seed 7	70.43
Seed 3	70.45	Seed 8	70.55
Seed 4	**70.83**	Seed 9	70.61
Seed 5	70.11	Seed 10	70.12

Table 12: Robustness of our model on different seeds for random initialization. The average of 10-models is 70.37%, and Std. the deviation is 0.253. Note that our official model is trained on seed 4.

But we choose the average score of the ensemble because the average ensemble is 1.28% higher than the maximum score ensemble in the test set.

Knowledge Distillation We optimize the weight option β of knowledge distillation. The final loss calculated when training knowledge distillation can be divided into two methods. The first method applies β only to the knowledge distillation loss term as $\mathcal{L} = \mathcal{L}_{ce} + \beta\mathcal{L}_{kd}$ in equation 7. The second method applies β to both terms, such as $\mathcal{L} = (1-\beta)\mathcal{L}_{ce} + \beta\mathcal{L}_{kd}$.

Figure 5 shows the optimization results for the hyper-parameter β used in the knowledge distillation when training with an ensemble knowledge distillation model. The experiment uses the loss function of equation 7 with methods and optimizes β between 0.1 and 1.0. When using the KLD, temperature (Hinton et al., 2015) is set to 5. As a result, the first method shows that the optimal performance is 71.18% F1 when β is 0.2 on the dev set. This method improves the F1 score by 0.34% compared to the single model. When β 0.1, F1 score is 71.06%, it is the second-best performance in the same method. In the case of the second method, when β is 0.3 and 0.5, F1 scores are 70.66% and 71.01%, respectively, which are improved than the single model. Accordingly, we can see that knowledge distillation of β below 0.5 is helpful for training, and it is meaningful to apply the loss of the first method to Korean coreference resolution.

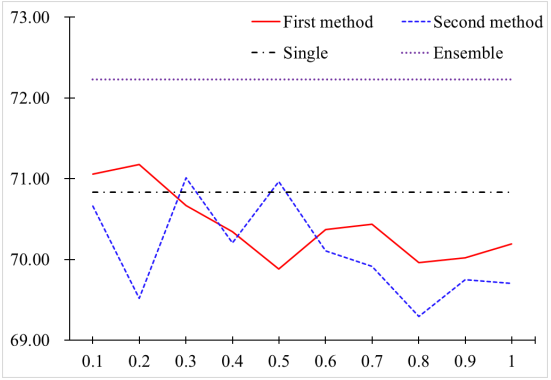

Figure 5: Hyperparameter β optimization of knowledge distillation on dev set of Korean coreference resolution
.