KNU CI System at SemEval-2018 Task4: Character Identification by Solving Sequence-Labeling Problem

Cheoneum Park[*], Heejun Song[**], Changki Lee[*]

[*]Department of Computer Science, Kangwon National University, South Korea
[**]Samsung Research, Samsung Electronics Co., Ltd., South Korea
[*]{parkce, leeck}@kangwon.ac.kr
[**]heejun7.song@samsung.com

Abstract

Character identification is an entity-linking task that finds words referring to the same person among the nouns mentioned in a conversation and turns them into one entity. In this paper, we define a sequence-labeling problem to solve character identification, and propose an attention-based recurrent neural network (RNN) encoder–decoder model. The input document for character identification on multiparty dialogues consists of several conversations, which increase the length of the input sequence. The RNN encoder–decoder model suffers from poor performance when the length of the input sequence is long. To solve this problem, we propose applying position encoding and the self-matching network to the RNN encoder–decoder model. Our experimental results demonstrate that of the four models proposed, Model 2 showed an F1 score of 86.00% and a label accuracy of 85.10% at the scene-level.

1 Introduction

In this paper, we define character identification (CI) (Chen et al., 2017) as a sequence-labeling problem and use a recurrent neural network (RNN) encoder–decoder (Enc–Dec) model (Cho et al., 2014) based on the attention mechanism (Bahdanau et al., 2015) to solve it. An Enc–Dec is an extension of the RNN model; it generates an encoding vector using an RNN in the encoder when an input sequence is given and performs decoding using the encoding vector. The attention mechanism calculates the alignment score for the two sequences and performs the input sequence and weighted sum so that they can focus more on the position that affects the output result. The self-matching network (Wang et al., 2017) is used to calculate an attention weight for itself and a context vector by using a weighted sum, after which the weights of similar words in the RNN sequence can be applied to aid in coreference resolution. Position encoding (PE) (Sukhbaatar et al., 2015, Park and Lee, 2017, Vaswani et al., 2017) is a method of applying weights differently, according to the positions of words appearing in a sequence. Training and prediction are performed by multiplying a weight vector by a vector of positions to be identified in a given input sequence.

In an Enc–Dec model, a long input sequence results in performance degradation due to loss of information in the front portion of the input sequence when encoding is performed. In this paper, we propose four models that apply PE, attention mechanism, and self-matching network to Enc–Dec models to solve the problem of performance degradation due to long input sequences.

To summarize, the main contributions of this paper are as follows:

1. In this paper, we define CI task as sequence-labeling problem, and perform training and prediction in end-to-end model.
2. We propose four models using Enc-Dec based on attention mechanism and achieve high performance.

2 System Description

An Enc–Dec model maximizes $P(y|x)$ using an RNN. The encoder generates an encoder hidden

Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018), pages 655–659
New Orleans, Louisiana, June 5–6, 2018.

state by encoding the input sequence, and the decoder generates an output sequence that maximizes $P(y|x)$ using the hidden state of the decoder, which was generated until this time step, with the encoder hidden state. The attention mechanism is a method of determining which part of the target class should be focused using the hidden state of the decoder and the hidden state of the encoder when performing decoding.

2.1 Model 1: Attention-based Enc–Dec model

The first model proposed in this paper is a general attention mechanism-based Enc–Dec model, as shown in Figure 1.

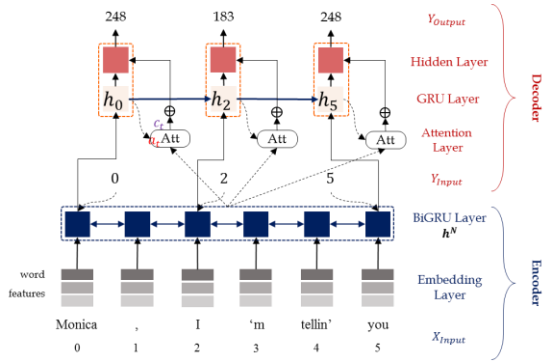

Figure 1: Attention-based Enc-Dec.

The input of the encoder is one document that contains S sentences (multiparty dialogue). Each sentence S consists of n_S words, and the input sequence X_{input} is $X_{input} = \{x_1, x_2, \dots, x_{n_S}\}$. The input to the decoder is $Y_{input} = \{yi_0, yi_1, \dots, yi_m\}$ consisting of the positions of the words given in the gold mentions, and the output sequence accordingly becomes $Y_{output} = \{yo_0, yo_1, \dots, yo_m\}$ consisting of the character number, which is corresponded with the decoder's input mentions.

We use word embedding and adopt the K-dimensional word embedding $e_i^k, k \in [1, K]$ for all input words, where i is the word index in the input sequence. We perform feature embedding for three features — speaker, named entity recognition (NER) tags, and capitalization — and concatenate them to make $\tilde{x}_i$.

- The uppercase feature is a binary feature (1 or 0) that verifies whether the uppercase is included in the word.
- 10-dimensional speaker embedding for a total of 205 different types of speakers included by "unknown".
- 19-dimensional NER embedding for a total of 19 different types of NER tags.

We use bidirectional gated recurrent unit (BiGRU) (Cho et al., 2014) for the encoder. The hidden state of the encoder for the input (word) sequence is defined as h_i^N.

$$e_i = W_e x_i \tag{1}$$

$$\tilde{x}_i = [e_i; uc_i; spk_i; NER_i] \tag{2}$$

$$h_i = biGRU(\tilde{x}_i, h_{i-1}) \tag{3}$$

where $\overrightarrow{h_i}$ and $\overleftarrow{h_i}$ are forward and backward networks, respectively, and h_i^N concatenates $\overrightarrow{h_i}$ and $\overleftarrow{h_i}$.

The decoder of our model uses the GRU as follows.

$$h_t = GRU(h_{yi_t}^N, h_{t-1}) \tag{4}$$

The input of the decoder is the hidden state h_i^N generated by the encoder corresponding to each position of Y_{input} which is the gold mention sequence. The hidden state h_t of the current decoder receives the hidden state h_i^N of the encoder corresponding to the output position of the previous decoder and the previous hidden state of the decoder.

$$\alpha_i^t = \frac{\exp\left(score_a\left(h_t, h_{yi_i}^N\right)\right)}{\sum_j \exp\left(score_a\left(h_t, h_{yi_j}^N\right)\right)} \tag{5}$$

$$score_a\left(h_t, h_{yi_i}^N\right) = \mathrm{v}_\mathrm{t}^T \tanh(W_a\left[h_t; h_{yi_i}^N; h_{yi_t}^N\right]) \tag{6}$$

$$y_t = \mathrm{argmax}_i(a_i^t) \tag{7}$$

$$c_t = \begin{cases} \sum_i a_i^t h_i^N, & soft\ attention \\ h_{y_t}^N, & hard\ attention \end{cases} \tag{8}$$

At the attention layer of the decoder, we use the attention weight α_i^t to compute the alignment score for the gold mention input into the decoder and the encoder hidden state h_i^N input. The attention layer acts as a coreference resolution for each gold mention and input sequence. After calculating the attention weights, we create the context vector c_t. We use soft attention and hard attention in Eq. (8). Soft attention $c_t = \sum_i a_i^t h_i^N$ is an attention-pooling vector of the whole input sentence of the encoder (h^N). The other attention-pooling vector is hard attention $c_t = h_{y_t}^N$, which is based on the argmax function Eq. (7) for attention weight α_i^t to choose the position with high score for the decoder input as the gold mention.

$$score_z(h_t, c_t, h_{yi_t}^N) = W_{z2}^T \mathrm{ReLU}(W_z[h_t; c_t; h_{yi_t}^N]) \tag{9}$$

$$yo_t = \text{argmax}_t\left(softmax\left(score_z\left(h_t, c_t, h^N_{yi_t}\right)\right)\right) \quad (10)$$

After calculating the context vector between the input of the encoder and the input of the decoder, we calculate $score_z$, using which the context vector c_t, decode hidden state h_t and encoder hidden state h^N are concatenated in the decoder hidden layer. Next, the softmax function is used to calculate the alignment score for $score_z$, and then the character index (Y_{output}) for the CI task corresponding to the input of the decoder is obtained using the argmax function.

2.2 Model 2: Attention–based Enc–Dec w/ model with PE

The second model is based on the first model but uses PE which is a method of applying a weight to an input sequence of an RNN according to the word order. Among the words in the coreference resolution, the antecedent has a feature that appears mainly in the preceding context. In this paper, we apply PE with a feature to the encoder input sequence, and use the weight according to the word order as the feature. As shown in Eq. (11), PE information is concatenated to Eq. (2) to produce $\tilde{x}_i$, and PE is calculated as shown in Eq. (12).

$$\tilde{x}_i = [e_i; PE_i; uc_i; spk_i; NER_i] \quad (11)$$

$$PE_i = (1 - i/n_s) - (s/k)(1 - 2i/n_s) \quad (12)$$

In PE, i is the index of the word, n_s is the total length of the input sequence, s is the position of the sentence, and k is the number of dimensions of the word expression. The weight of PE is calculated as a real value that gradually decreases between 1 and 0, and is applied to the input of the encoder to take advantage of the feature that the predecessor precedes the current mention. In Eq. (12), $(1 - 2i/n_s)$ denotes the order of words. If it is a front word, it has a higher value than the next word. (s/k) is a weight based on the sentence order, and when the sentence is different, the weight reduction rate difference is calculated to be higher than the value decreasing in the sentence. The expression for the encoder and decoder are the same as for model 1.

2.3 Model 3: Self-matching Network-based Enc-Dec model

The third model is also based on the first model, but performs encoding by using the self-matching network in the encoder without using PE, as shown in Figure 2. The self-matching network is used for calculating the alignment score for a given RNN sequence and itself, and then for performing a weighting sum with itself to create a context vector. While using the self-matching network for encoding, attention weights are weighted with high alignment scores between similar words. For example, if "Rachel's child-hood best friend" and "Monica" appear in a sentence, a high alignment score between them is calculated by the self-matching network.

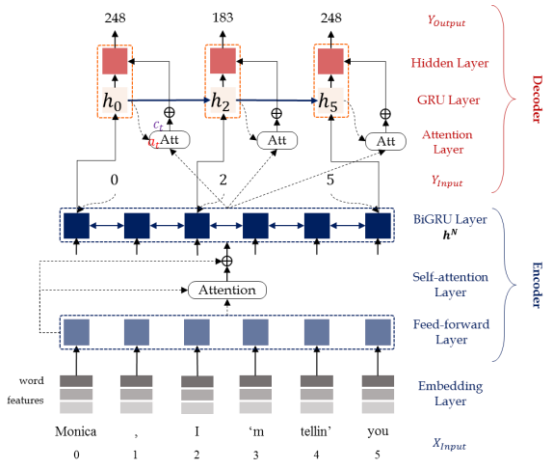

Figure 2: Self-matching–Network-based Enc–Dec.

The input sequence of the encoder becomes $\tilde{x}_i$ in Eq. (2), and a feed-forward neural network is used, as in Eq. (13). Next, we use the self-matching network to compute the attention weight for the t sequence and create a context vector c_i^{self} that reflects the self-attention (Eqs. 14–16).

$$h_i^{src} = W_{src}\tilde{x}_i + \mathrm{b}_{src} \quad (13)$$

$$\alpha_j^i = \frac{\exp\left(score_{self}\left(h_j^{src}, h_i^{src}\right)\right)}{\sum_t \exp\left(score_{self}\left(h_t^{src}, h_i^{src}\right)\right)} \quad (14)$$

$$score_{self}\left(h_j^{src}, h_i^{src}\right) = v_i^T \tanh(W_a^{src}\left[h_j^{src}; h_i^{src}\right]) \quad (15)$$

$$c_i^{self} = \sum_j \alpha_j^i h_j^{src} \quad (16)$$

Subsequently, we construct $\left[h_i^{src}; c_i^{self}\right]$ by concatenating the context vector c_i^{self}, created using the self-matching network, and the hidden state h_i^{src}; it is then fed to the input of the hidden layer of the encoder to perform BiGRU.

We apply $\left[h_i^{src}; c_i^{self}\right]$ to the additional gate to make $\left[h_i^{src}; c_i^{self}\right]^*$, this determines whether or not

the information of $[h_i^{src}; c_i^{self}]$ is transmitted to the encoder hidden layer input. The equation is as follows:

$$g_t = sigmoid(W_g[h_i^{src}; c_i^{self}]) \quad (17)$$

$$[h_i^{src}; c_i^{self}]^* = g_t \odot [h_i^{src}; c_i^{self}] \quad (18)$$

The decoder of model 3 performs training and prediction using a decoder such as the one used in model 1 (Eqs. 4–8) based on the hidden state where encoding is performed as above.

2.4 Model 4: Self–matching Network–based RNN Enc-Dec model with PE

Model 4 is based on model 3 using the self-matching network; it additionally uses PE, which was also used for model 2, as a feature to confirm the word order.

3 Experimental Results

We evaluate the entity linking performance of the models using label accuracy and macro-F1 (Chen et al., 2017), and the coreference resolution performance using CoNLL F1 (Rahman and Ng, 2009).

The word representation used in this paper is a data–set provided by LDC, which is learned by a neural network language model (Bengio et al., 2003, Lee et al., 2014), and is set to 50 dimensions. The experiments were performed with cross validation. The hyper parameters used in the experiment are as follows. We used tanh for the encoder and decoder, and ReLU for the attention layer. The hidden layers had 150 dimension, and the dropout of all layers was set to 0.3. The learning was done using RMSprop (Hinton et al., 2012) and the learning rate was reduced by 50% for every 5 epochs without performance improvement starting at 0.1. The decoder attention functions of the models used in the experiments are all based on hard attention, and are compared with soft attention in Table 1.

	Episode-Level		Scene-Level	
	Main F1	**All F1**	**Main F1**	**All F1**
M 1	85.57	**23.33**	83.45	21.13
M 2	82.91	23.15	85.58	22.40
M 2'	82.46	22.04	83.56	19.67
M 3	**86.30**	22.93	84.10	21.84
M 4	83.65	22.13	**87.41**	22.06
M 4'	85.48	19.46	86.23	**23.92**

Table 1: Entity-linking results on the trial set (in %). Main and All in column mean main and other characters, and all characters.

Table 1 shows a comparison between the CI performances of the models on the trial set. M 2' is a model in which PE proposed by Vaswani et al. (2017) is applied, and M 4' is a model in which soft attention is applied to M 4. At the episode-level, M 3 showed the best Main F1 performance (86.30%) and M 1 showed the best All F1 performance (23.33%). At the scene-level, M 4 showed the highest Main F1 performance (87.41%), and M 4 showed the highest All F1 performance (23.92%). In the case of M 2 and M 2', we can see that the proposed PE method resulted in a better overall performance.

At the episode-level, M 4' showed a better Main F1 performance (1.83%) than M 4, whereas M 4 showed a better All F1 performance (by 2.67%). At the scene-level, M 4 showed a better Main F1 performance (by 1.18%) than M 4', whereas M 4'

		Characters							Main + Other		All	
Model		**Chandler**	**Joey**	**Monica**	**Phoebe**	**Rachel**	**Ross**	**Others**	**F1**	**Acc**	**F1**	**Acc**
E	M 1	81.17	76.39	86.86	84.70	87.13	81.03	72.86	81.45	80.32	15.75	67.19
	M 2	85.77	81.59	85.19	87.67	59.64	84.79	80.42	**85.01**	**84.36**	16.47	**68.42**
	M 3	82.76	82.06	86.36	84.21	88.76	81.78	77.40	83.33	82.38	**17.02**	66.65
	M 4	83.52	79.92	87.17	86.43	86.72	83.73	78.46	83.71	82.96	15.19	65.83
S	M 1	83.55	79.19	90.99	86.43	90.01	84.11	76.08	84.34	83.08	15.93	68.59
	M 2	84.94	79.67	91.16	88.09	92.49	85.86	79.79	**86.00**	**85.10**	**16.98**	**69.49**
	M 3	83.87	84.30	88.24	86.10	88.79	79.65	76.23	83.88	82.34	16.99	67.31
	M 4	78.21	83.06	84.42	84.30	89.94	79.08	90.41	85.33	84.64	15.43	67.68
Amore		-	-	-	-	-	-	-	79.36	77.23	**41.05**	**74.72**
Kamp.		-	-	-	-	-	-	-	73.51	73.36	37.37	59.45
zuma		-	-	-	-	-	-	-	43.15	46.07	14.42	25.81

Table 2: Entity-linking results on the evaluation set (in %). The F1 score is reported for each character. E/S: episode/scene level. F1 is macro-average F1 score. Acc is character label accuracy.

showed a better All F1 performance (by 1.86%). Thus, it can be seen that the use of hard attention results in a better performance.

Table 2 presents the experimental results of the test set (episode- and scene-level) for the method proposed in this paper, and the performance comparison with other competing models, namely AMORE UPF (Amore), Kampfpudding (Kamp.), and zuma. In the Main + Other character evaluations at episode-level, M 2 showed the best performance among all models (F1 of 85.01%, Acc of 84.36%), whereas in the All character evaluations, M 3 showed the best F1 performance (17.02%) and M 2 showed the best Acc performance (68.42%). At the scene-level, M 2 showed the best performance in both the Main + Other and the All character evaluation. The proposed method showed a lower overall performance in the All character evaluation compared with other competing models, but showed a higher performance in the Main + Other character evaluations. The reason for the lower performance in the All character evaluation is that the number of data points is smaller than that of the main characters.

4 Conclusion

In this paper, we defined the entity-linking problem of SemEval-2018 Task 4 as a sequence-labeling problem and proposed four models to solve it. Experimental results showed that M 2 shows the best performance in the test set scene-level (Main + Other characters), with an F1 of 86.00% and Acc of 85.10%. In the Main entities + Others evaluation of SemEval-2018 Task 4, it ranked 1st with an F1 of 83.37% and Acc of 82.13%. In All Entities + Others, it ranked 2nd with an F1 of 13.53% and Acc of 68.55%.

In future work, we will apply character CNN to solve the unknown word problem, and we will add word expressions such as GloVe (Pennington et al., 2014) and ELMo (Peters et al., 2018). We will also enhance the performance by tightening the model with less data by adding the features used in the task 4-based model.

References

Henry Y. Chen, Ethan Zhou, and Jinho D. Choi. 2017. Robust Coreference Resolution and Entity Linking on Dialogues: Character Identification on TV Show Transcripts. *In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv:1406.1078*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *Proc. of ICLR' 15*, *arXiv:1409.0473*.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated Self-Matching Networks for Reading Comprehension and Question Answering, *In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* pages 189-198.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-To-End Memory Networks. *arXiv:1503.08895*.

Cheoneum Park and Changki Lee. 2017. Korean Coreference Resolution using Pointer Networks based on Position Encoding. *In: Proceedings Of the KIISE and the KBS Joint Symposium*, pages 76-78.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *arXiv:1706.03762*.

Altaf Rahman and Vincent Ng. 2009. Supervised Models for Coreference Resolution. *In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP),* pages 968-977.

Changki Lee, Junseok Kim, and Jeonghee Kim. 2014. Korean Dependency Parsing using Deep Learning. *In: Proceedings of the KIISE for HCLT*, pages 87-37.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A Neural Probabilistic Language Model. *Journal of machine learning research,* pages 1137-1155.

Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. 2012. Rmsprop: Divide the gradient by a running average of its recent magnitude. *Neural networks for machine learning, Coursera lecture 6e*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the 2014 conference on empirical methods in natural language processing* (EMNLP), pages. 1532-1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word prepresentations. *arXiv:1802.05365*.