

An Efficient Approach based on BERT and Recurrent Neural Network for Multi-turn Spoken Dialogue Understanding

Weixing Xiong^{*a}, Li Ma^{*b} and Hongtao Liao

Ubtech Robotics Corp, Nanshan I Park, No.1001 Xueyuan Road, Shenzhen, China

Keywords: BERT, Recurrent Neural Network, Multi-turn, Spoken Dialogue Understanding.

Abstract: The main challenge of the Spoken Language Understanding (SLU) is how to parse efficiently natural language into effective meanings, such as its topic intents, acts and pairs of slot-values that can be processed by computers. In multi-turn dialogues, the combination of context information is necessary to understand the user's objectives, which can be used to avoid ambiguity. An approach processing multi-turn dialogues, based on the combination of BERT encoding and hierarchical RNN, is proposed in this paper. More specifically, it combines the current user's utterance with each historical sequence to formulate an input to the BERT module to extract the semantic relationship, then it uses a model derived from the hierarchical-RNN for the understanding of intents, actions and slots. According to our experiments by testing with multi-turn dialogue dataset Sim-R and Sim-M, this approach achieved about 5% improvement in FrameAcc compared with models such as MemNet and SDEN.

1 INTRODUCTION

In a task-oriented dialogue, all the information needed for a specific task or purpose may not be given in a single turn expression by the user's utterance. It's necessary to engage a multi-turn dialogue to obtain mandatory information. The main function of a SLU module is to extract efficiently, from the user's input, the intents, actions and slots-value pairs (Dilek Hakkani-Tur et al., 2016).

```
U1: I need to make reservation for 6 pm.
S1: What date and restaurant would you like?
U2: Next monday at 11 Fornaiio .
      ↓      ↓      ↓      ↓      ↓
Slot: B-date I-date 0 B-rest I-rest 0
Intent: reserve_restaurant
Dialogue Acts: inform(date=next monday),
              inform(rest=11 Fornaiio)
```

Figure 1: An example semantic frame with slot, intent and dialogue act annotations, following the IOB tagging scheme.

Figure 1 shows the semantic frame information about a task-oriented dialogue, in which the word slot is represented in a general IOB format.

In real dialogue, all necessary information will be specified along with the dialogue flow. Let's continue the above example:


S2: "How many people will attend the dinner?"


U3: "5."

So, the user utterance, "5", corresponds to the entity category "B-#people", for the circumstance of the booking restaurant task. The difficulty is how a computer system can analyze all the information to extract the intents, acts and slots.

Such as the one proposed by P. Xu and R. Sarikaya, 2013, B. Liu et al., 2016, or Zhang et al., 2016, use the method of jointly modeling intents and slots with RNN, but they do not take the necessary context information into account.

MemNet, proposed by S. Sukhbaatar et al., 2015; Chen et al., 2016, or SDEN, proposed by Ankur Bapna et al., 2017, Raghav et al., 2018, can effectively take into account the historical context to understand semantic frames by encoding information as

^a <https://orcid.org/0000-0002-0929-096X>

^b <https://orcid.org/0000-0002-9297-4632>

*Equal contribution, ordered by random shuffle.

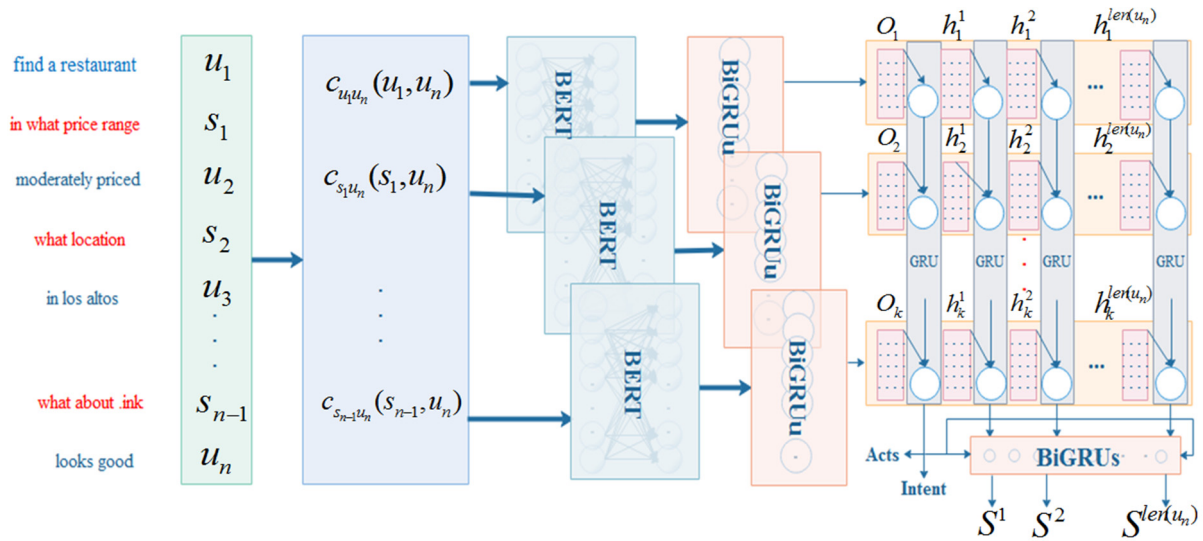


Figure 2: MSDU model. The model contains two kinds of RNN models, $RNN(BiGRU_u)$ encoding over utterance tokens while $RNN(BiGRU_g$ and $BiGRU_s)$ encoding over utterances.

sentence vectors by GRU. However, in MemNet and SDEN, each sentence needs to be encoded as a single vector, this leads to the loss of lexical-granularity information when analysing the relationship between the current sentence and the historical inputs.

In this paper, we propose a model based on BERT¹ (Jacob Devlin et al., 2018) and hierarchical RNN.

We encode the current user’s utterance and historical dialogues successively as the input of BERT module, and then to encode the memory from context, we use a modified hierarchical-RNN to process the outputs of BERT module.

There are three main aspects of our contribution. Firstly, by using the BERT model, the attention of adjacent words is introduced for word and sentence embedding. Secondly, by concatenating the current utterance with each historical utterance, the model can calculate attention with other turns of utterance when performing BERT encoding. Thirdly, by using a modified hierarchical-RNN to process the outputs of BERT module, information from the context can be more effectively encoded.

The following sections in this paper are organized as: In section 2, we describe the general architecture of our model. In section 3, we list the experimental results and analyze them. In last section, the conclusion and discussion would be illustrated.

¹ The open source BERT implementation based on Pytorch is available at <https://github.com/huggingface/pytorch-pretrained-BERT>. Note that the pre-trained BERT has two version. In our experiment, we use the base version.

2 MSDU MODEL

We abbreviated our new model to MSDU, which is an acronym for “Multi-turn Spoken Dialogue Understanding”. The model we proposed is dedicated to handle multi-turn spoken dialogue understanding and intents information extraction, its overall structure is shown in Figure 2. We divide a sequence of dialogues into n turns, and each of them containing the user’s utterance u_n and the system responding utterance s_n . The user’s current (the last input) utterance u_n can be represented by formula (1).

$$u_n = \{w_n^1, w_n^2, w_n^3, \dots, w_n^{\text{len}(u_n)-1}, w_n^{\text{len}(u_n)}\} \quad (1)$$

Where w_n represents word token in the utterance and $\text{len}(u_n)$ means the number of tokens in u_n .

So, there are $n-1$ user utterances and $n-1$ system replies in the historical dialogue, represented as formula (2).

$$D = \{u_1, s_1, u_2, s_2, \dots, u_{n-1}, s_{n-1}\} \quad (2)$$

In multi-turn dialogues, the important matter is how to use effectively the context information in the conversation to track the current state. In order to obtain the related information between the current user utterance and the historical utterances, we need to build some relationship between them. Therefore, we

have designed a concatenation method, its detail is given in the Part 2.1.

2.1 Concatenation Method

We concatenate the current user utterance u_n with each utterance in the historical turns $(u_1, s_1, u_2, s_2, \dots, u_{n-1}, s_{n-1})$ to form a new couple sentence vector C . For instance, the concatenation of (u_n, s_{n-1}) is expressed as followings:

$$C_{s_{n-1}u_n} = \{[CLS], w_{s_{n-1}}^1, \dots, w_{s_{n-1}}^{\text{len}(s_{n-1})}, [SEP], w_{u_n}^1, \dots, w_{u_n}^{\text{len}(u_n)}, [SEP], [PAD], \dots, [PAD]\} \quad (3)$$

$$B_{s_{n-1}u_n} = \{\underbrace{0, \dots, 0}_{\text{len}(s_{n-1})+2}, \underbrace{1, \dots, 1}_{\text{len}(u_n)}, 0, \dots, 0\} \quad (4)$$

$$C_{u_1u_n} = \{[CLS], w_{u_1}^1, \dots, w_{u_1}^{\text{len}(u_1)}, [SEP], w_{u_n}^1, \dots, w_{u_n}^{\text{len}(u_n)}, [SEP], [PAD], \dots, [PAD]\} \quad (5)$$

$$B_{u_1u_n} = \{\underbrace{0, \dots, 0}_{\text{len}(u_1)+2}, \underbrace{1, \dots, 1}_{\text{len}(u_n)}, 0, \dots, 0\} \quad (6)$$

Where w_u^i indicates the i -th token in utterance u , $\text{len}(u)$ and $\text{len}(s_{n-1})$ means respectively the number of tokens in utterance u and s_{n-1} . $[CLS]$, $[SEP]$ and $[PAD]$ are special tags in BERT inputs, where $[CLS]$ represents the beginning of a sequence, $[SEP]$ is a separator for two sequences, and $[PAD]$ is used to pad all sequences to the same length.

In order to facilitate the calculation of the model, we need to add paddings in the sequence to a fixed number, in our experiment case, it is set to 64.

Then we generate a Boolean vector to indicates which words are from the current user utterance in C . The generated Boolean vector B is shown in formula (4) and formula (6). In our example, we obtain $64 - \text{len}(u_n)$ of 0 and $\text{len}(u_n)$ of 1.

In real application scenario, the user's current utterance could not only be a response to the last system utterance, but also a response to an earlier system utterance or a supplement to previous user utterance.

Therefore, in order to get a better modeling effect, we need to take all historical utterances from u_1 to s_{n-1} into account, rather than just concatenating u_n with last system utterance s_{n-1} . The tricks are as formula (5-6).

In this way, the results shown in formula (7) can be obtained one by one. Then we can obtain $2*(n-1)$ pairs.

$$inputs = \underbrace{\{(C_{u_1u_n}, B_{u_1u_n}), \dots, (C_{s_{n-1}u_{n-1}}, B_{s_{n-1}u_{n-1}})\}}_{2*(n-1)} \quad (7)$$

We will then use two types of RNN to further process the concatenated utterance pairs. The first one is used to encode the relationship between words in single utterance pair, calling tokens-level RNN; the second one is used to integrate information about all utterance pairs, calling utterances-level RNN.

2.2 RNN over Utterance Tokens

In order to get the relationship between word tokens in the dialogue utterance, we feed the pre-trained BERT model with the concatenated pairs (C, B) . The outputs are represented by H with k vectors of $64*768$ dimension, in formula (8), where, k is the pairs of the historical utterances, 64 is the sequence number after padding, 768 is the default size of hidden layer in BERT's outputs.

$$\left. \begin{aligned} \mathbf{H}_1 &= \text{BERT}(C_{u_1u_n}, B_{u_1u_n}) \\ \mathbf{H}_2 &= \text{BERT}(C_{s_1u_n}, B_{s_1u_n}) \\ &\vdots \\ \mathbf{H}_k &= \text{BERT}(C_{s_{n-1}u_n}, B_{s_{n-1}u_n}) \end{aligned} \right\} k \quad (8)$$

We use the BASE version of pre-trained BERT in our test, and the parameters of BERT model are fixed during training, for the concern of computing speed. The outputs of BERT model are introduced into the tokens-level RNN, which is a BiGRU model, for fine tuning. Specific encoding results are shown as formula (9):

$$\left. \begin{aligned} (\mathbf{o}_{1f}^1, \mathbf{o}_{1f}^2, \dots, \mathbf{o}_{1f}^l, \mathbf{o}_{1b}^1, \mathbf{o}_{1b}^2, \dots, \mathbf{o}_{1b}^l) &= \text{BiGRU}_u(\mathbf{H}_1) \\ (\mathbf{o}_{2f}^1, \mathbf{o}_{2f}^2, \dots, \mathbf{o}_{2f}^l, \mathbf{o}_{2b}^1, \mathbf{o}_{2b}^2, \dots, \mathbf{o}_{2b}^l) &= \text{BiGRU}_u(\mathbf{H}_2) \\ &\vdots \\ (\mathbf{o}_{kf}^1, \mathbf{o}_{kf}^2, \dots, \mathbf{o}_{kf}^l, \mathbf{o}_{kb}^1, \mathbf{o}_{kb}^2, \dots, \mathbf{o}_{kb}^l) &= \text{BiGRU}_u(\mathbf{H}_k) \end{aligned} \right\} k \quad (9)$$

In the above equations, each \mathbf{o} with subscript f is the result hidden layer of BiGRU forward propagation calculation, and each \mathbf{o} with subscript b is the result of BiGRU backward propagation calculation, in our experiment, the size of o is set to 64, and l is the number of input sequence after padding set to 64.

2.3 RNN over Utterance Context

When parsing the concatenated utterance pairs, we consider following formula (10):

$$\begin{aligned} \mathbf{h}_1^1, \dots, \mathbf{h}_1^{\text{len}(u_n)} &= f(\hat{\mathbf{B}}_{u_n} \times [\mathbf{o}_{1f}^1 \oplus \mathbf{o}_{1b}^1; \dots; \mathbf{o}_{1f}^1 \oplus \mathbf{o}_{1b}^1]) \\ \mathbf{h}_2^1, \dots, \mathbf{h}_2^{\text{len}(u_n)} &= f(\hat{\mathbf{B}}_{s_i u_n} \times [\mathbf{o}_{2f}^1 \oplus \mathbf{o}_{2b}^1; \dots; \mathbf{o}_{2f}^1 \oplus \mathbf{o}_{2b}^1]) \\ &\vdots \\ \mathbf{h}_k^1, \dots, \mathbf{h}_k^{\text{len}(u_n)} &= f(\hat{\mathbf{B}}_{s_n u_n} \times [\mathbf{o}_{kf}^1 \oplus \mathbf{o}_{kb}^1; \dots; \mathbf{o}_{kf}^1 \oplus \mathbf{o}_{kb}^1]) \end{aligned} \quad (10)$$

Where $\hat{\mathbf{B}}_{u_n}$ and $\hat{\mathbf{B}}_{s_i u_n}$ are diagonal matrix generated from Boolean vectors built from formula (6), those diagonal matrix have size 64×64 . \oplus is the concatenate operator, the size of \mathbf{o} is set to 64, and the size of those concatenated vectors will be 128. Vectors wrapped with bracket make up a matrix, the elements in the bracket represent the row vectors that make up the matrix. In our experiments, the concatenated matrix size is 64×128 . f is the selection operator used to fetch out all non-zero row vectors from a matrix. It is easy to deduce that the number of non-0 row vectors of each matrix is equal to the number of non-0 elements on the diagonal line of $\hat{\mathbf{B}}$, which equals to $\text{len}(u_n)$, the number of tokens in the utterance u_n , so the fetching operation finally allow us to obtain $\text{len}(u_n)$ row vectors. Each of these vectors represents the embedding of a special word in the current user utterance, with the attention information.

The outputs of formula (10) are then used as the inputs of the utterances-level RNN to encode the slot-tag of each tokens in user utterance.

The prediction of intents and actions needs different information on context from that of slots. The utterances-level RNN used for the prediction of intents and actions does not share parameters whereas it does for that of slots. In each row in formula (9), the last hidden layer in both directions is concatenated to encode information for the whole utterances pair. In this way, we proceed a concatenation of (O_f, O_b) to get vectors O_n ($n=1, \dots, k$) for full text understanding, shown in formula (11).

$$\left. \begin{aligned} \mathbf{O}_1 &= \mathbf{o}_{1f}^1 \oplus \mathbf{o}_{1b}^1 \\ \mathbf{O}_2 &= \mathbf{o}_{2f}^1 \oplus \mathbf{o}_{2b}^1 \\ &\vdots \\ \mathbf{O}_k &= \mathbf{o}_{kf}^1 \oplus \mathbf{o}_{kb}^1 \end{aligned} \right\} k \quad (11)$$

Then we take all vectors from O_1 to O_k as inputs to the utterances-level RNN model, and take the last hidden layer as context embedding, which is expressed in formula (12).

$$\mathbf{G} = \text{GRU}_g(\mathbf{O}_1, \mathbf{O}_2, \dots, \mathbf{O}_{k-1}, \mathbf{O}_k) \quad (12)$$

Note that in formula (12), the output \mathbf{G} used is the last hidden layer of GRU_g , which gives the classification or prediction of intents and acts.

For the prediction of slots, the selected attention distribution is used as input to the utterances-level RNN, given by the formula (13).

$$\mathbf{S}_j = \text{GRU}_s(\mathbf{h}_1^j, \mathbf{h}_2^j, \dots, \mathbf{h}_{k-1}^j, \mathbf{h}_k^j) \quad (13)$$

In formula (13), \mathbf{S}_j is the last hidden layer of GRU_s , its output gives information for slot prediction corresponding to the j -th token in the current user utterance, and j satisfies $1 \leq j \leq \text{len}(u_n)$.

Thus, the named-entity information of each word in current utterance can be obtained with the attention information taking into account of the dialogue context, as shown in formula (14):

$$\mathbf{S} = \{\mathbf{S}_1, \mathbf{S}_2, \dots, \mathbf{S}_{\text{len}(u_n)-1}, \mathbf{S}_{\text{len}(u_n)}\} \quad (14)$$

The \mathbf{G} and \mathbf{S} obtained above are used for the determination of intents, acts and slots, computed as formula (15), (16) and (17), the same as that used by MemNet and SDEN.

$$\mathbf{P}^{\text{Intent}} = \text{Softmax}(\mathbf{U}\mathbf{G}) \quad (15)$$

$$\mathbf{P}^{\text{Act}} = \text{Sigmoid}(\mathbf{V}\mathbf{G}) \quad (16)$$

Inspired by memory network and SDEN, we take the value of \mathbf{S} as the inputs of another BiLSTM model, take the value of \mathbf{G} as the original hidden layer $h(0)$ of this model, Then we put the result of its output layer into the Softmax layer to get the named-entity prediction of each word in the current utterance.

$$\mathbf{P}_i^{\text{Slot}} = \text{Softmax}(\text{BiLSTM}(\mathbf{S} | \mathbf{G})_i) \quad (17)$$

In formula (17), $\mathbf{P}_i^{\text{Slot}}$ means the estimated probability vector of the i -th word in current utterance, each element in the vector represents the probability that the i -th word belongs to the corresponding entity category. In the expression $\text{BiLSTM}(\mathbf{S} | \mathbf{G})_i$, i means the i -th hidden layer of BiLSTM as output, the parameter \mathbf{G} coming from formula (12) means initial hidden layer of BiLSTM.

We compute the loss based on cross-entropy for each sub-task, take the sum of them as the total loss, and we optimize our model based on the total loss.

3 EXPERIMENTS

In order to verify the effectiveness of the model, we use the dataset Sim-R and Sim-M for training and test, the same preformed with those used by MemNet and SDEN.

3.1 Dataset

The datasets Sim-R and Sim-M (Shah P et al. 2018) are widely used in context-based intents, acts and slots joint recognition tasks.² Sim-R is a dataset in multi-turn conversation for the restaurant domain, the training set contains 1116 dialogues, 11234 interactions; and Sim-M is a dataset in multi-turn conversation for the movie domain, the training set contains 384 dialogues, 3562 interactions. Table 1 gives a glance at this dataset.

We combine together two training sets respectively into one for the training, and then test the model using uncombined test set and combined test set. In some cases, when user utterances appear at the beginning of the dialogue without act labels, the corresponding act label is set to be "OTHER".

3.2 Baselines

For benchmark, we use a 2-layer RNN model without considering the context information, and also MemNet and SDEN. For MemNet and SDEN.

In order to study the effectiveness of each component of the proposed model MSDU, we made survey specific ablation tests. In the first case, we remove both the BERT module and the concatenate process. In the second case, the BERT module was replaced by random initialized words embedding. In the third case, we don't concatenate the sentence explained above for the BERT module. And finally, we proceed the concatenation of sentences with the BERT module. We used also CRF module with MSDU for slots recognition. In the following, we give a more detailed explanation above the different models used for comparison.

NoContext: Regardless of dialogue information from context, the model's structure is the same as the current utterance processing module in MemNet and SDEN. A two-layer RNN structure consist of one GRU and one LSTM is adopted. The difference is that

the initial hidden layer of LSTM is all-zero vector, so it does not contain any information about context.

MemNet: The attention scores of current sentence vectors and historical sentence vectors are calculated based on cosine similarity, context vector is the sum of historical sentence vectors weighted by their attention scores. The current sentence processing module is the same as **NoContext**, and the word embedding is randomly initialized.

MemNet+FastText: The MemNet model used with word embedding matrix initialized with 300-dimensional pre-trained word embedding from FastText³ (E. Grave, P. Bojanowski et al. 2018).

SDEN: It is a modification of MemNet with randomly initialized word embedding. It calculates the attention between the current utterance vector and the historical ones using a linear full connection layer, and inputs the attention vectors into a GRU model for obtaining the context vector.

SDEN+FastText: It is a SDEN model with the word embedding matrix initialized with 300-dimensional pre-trained word vectors FastText.

MSDU-BERT-Concat: This MSDU model does not use BERT module nor concatenation process, it uses hierarchical-GRU to encode context memory.

MSDU-BERT: In this case, a random initialized 300-dimensional words embedding matrix is used instead of BERT module.

MSDU-Concat: No dialogue utterances concatenation used with the MSDU model, we embedded each historical utterance into a 300-dimensional vector using BERT-GRU. Then all these vectors are embedded into a single 300-dimensional vector to GRU to characterize context memory. Further, we use this vector as first hidden state of GRU when processing current utterance.

MSDU+CRF: It is a MSDU, in the process of prediction, we use Viterbi algorithm to solve the sequence with the highest probability.

3.3 Training and Evaluation

The hyperparameters of all models are set as follows:

- Batch_size: 64
- Dropout ratio: 0.3
- Word embed size: 64
- Hidden size for sent encoding: 64

² This dataset could be downloaded from <http://github.com/google-research-datasets/simulated-dialogue>

³ Available at <https://github.com/facebookresearch/fastText/blob/master/docs/crawl-vectors.md>.

Table 1: Profile of datasets used in the experiments, with values of intents, acts, slots, and number of dialogues (A dialogue may include many turns of interactions between user and system).

Dataset	Intents	Acts	Slots	No.Train	No.Dev	No.Test
Sim-R	FIND_RESTAURANT, RESERVE_RESTAURANT	THANK_YOU, INFORM, AFFIRM, CANT_UNDERSTAND, REQUEST_ALTS, NEGATE, GOOD_BYE, OTHER	price_range, location, restaurant_name, category, num_people, date, time	1116	349	775
Sim-M	BUY_MOVIE_TICKETS	OTHER, GREETING, GOOD_BYE, CANT_UNDERSTAND, THANK_YOU, NEGATE, AFFIRM, INFORM	theatre_name, movie, date, time, num_people	384	120	264

Table 2: SLU results on test sets with baselines and MSDU, when trained on Sim-M + Sim-R, "Overall" means the test set is Sim-R + Sim-M. Because any of the above models can add CRF module, we did not consider MSDU+CRF when marking the maximum value using bold print.

Model	Intent F1			Act F1			Slot F1			FrameAcc		
	Sim-R	Sim-M	Overall	Sim-R	Sim-M	Overall	Sim-R	Sim-M	Overall	Sim-R	Sim-M	Overall
NoContext	82.04	68.47	78.33	88.37	88.74	88.48	97.56	94.70	96.64	71.13	45.89	63.96
MemNet	99.82	98.39	99.44	94.97	89.35	93.38	97.64	94.00	96.56	86.90	65.90	80.88
MemNet+FastText	99.50	99.63	99.56	91.80	89.56	91.18	97.39	94.44	96.55	83.76	67.23	79.13
SDEN	98.08	98.75	98.35	92.66	87.50	91.16	97.59	94.21	96.59	85.36	65.76	79.29
SDEN+FastText	99.71	99.85	99.73	89.26	90.65	89.61	97.39	94.78	96.59	83.56	69.06	79.44
MSDU-BERT-Concat	99.88	99.93	99.90	96.57	92.16	95.27	96.55	93.86	96.55	88.10	67.67	82.29
MSDU-BERT	99.80	99.93	99.85	96.40	90.58	94.75	97.68	95.84	97.13	87.25	73.53	83.40
MSDU-Concat	99.50	99.93	99.62	96.89	91.82	95.39	98.20	97.22	97.90	88.32	76.76	85.02
MSDU	99.85	99.93	99.88	96.94	92.16	95.56	98.01	97.33	97.81	88.68	78.30	85.73
MSDU+CRF	99.88	99.93	99.90	97.20	92.30	95.76	98.00	97.09	97.75	89.26	78.30	86.19

For all models, we use the same ADAM optimizer. The initial learning rate is set to 0.001, which decreases to 0.0001 after 125th epoch and 0.00001 after 250th epoch, $\text{betas}=(0.9, 0.999)$, $\text{eps}=1e-8$. The results are evaluated for verification set on every epoch. The model saved when the FrameAcc breaks the historical record, and the training process is terminated after 500epoches. We used the last saved model for test.

Figure 3 shows the main performance rates of MSDU with the training epochs. The F1-score of intention recognition reached a high level even at the first epoch, and the performance curves stay at their highest level as soon as the 10th epoch.

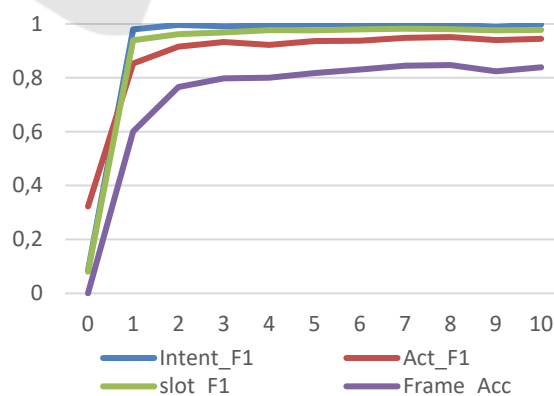


Figure 3: Main performance measures on evaluation set change with the training epochs.

3.4 Results and Analysis

Table 2 shows the results of each model with the respectively intents, acts and slots recognition. The last column of FrameAcc shows the proportion of correct recognition of intents, acts and slots for each model used in the test experiment. The second row lists the test data set used, and “Overall” represents the new test set combination of Sim-R and Sim-M.

For MemNet and SDEN, we find that the model using random initialized word embedding gives better performance on sim-R dataset with larger sample size. However, with the sim-M dataset with smaller sample size, the model with the pre-trained word embedding is more satisfied.

For the recognition of intent, the model NoContext is significantly worse than all other models. This can explain that the task of intent recognition is more dependent on context. Due to the introduction of contextual information, all other models obtain high accuracy in intent recognition. MSDU model achieves obviously the best results compared with other models.

For the task of act recognition, the performance of NoContext is still lower than other models, which proves that the information from context is still helpful. The performance of MSDU in the act recognition is obviously better than that of other models, that means MSDU has a stronger ability in understanding the relationship between context and the current user utterance.

For the recognition of slot tagging, there is no significant difference of performance for the Models MemNet, SDEN and NoContext. In the other hand, MSDU and its variant models achieve better results. At the same time, we also find that MSDU-Concat is nearly the same as MSDU for slot recognition, meaning that the concatenation process is not very useful for slot recognition improvement.

From the test results, we find that the MSDU model achieves about 5% better for FrameAcc than MemNet and SDEN models.

It is interesting to notice that SDEN does not obtain a better result than MemNet even although the forth one using a more complex context encoding method. MSDU-BERT-Concat and above two models use the same random initialized word embedding method. The difference lies mainly in term of context encoding: the model MSDU-BERT-Concat uses a hierarchical-GRU to encode context information, which is even simpler than the context encoding method used by MemNet, however it obtains about 2% better for FramAcc than MemNet and SDEN.

This causes a doubt for the necessity of attention mechanism in context encoding.

From the results produced by the MSDU variant models, we can also conclude that the concatenation procedure brings about 1.1% of improvement, the BERT module brings about 2.7%, and the combination of the both gives 3.4% of improvement.

4 CONCLUSIONS AND FUTURE WORKS

The MSDU model is proposed for the recognition of intents, acts and slots with the historical information in a multi-turn spoken dialogue through training with different datasets and variant modification. The test result shows that the design concept of MSDU model is more effective and brings important improvement.

For future works, we will study how to apply this new model architecture for higher level dialogue understanding tasks, such as ontology-based slot recognition, and the alignment of intent-act-slot. For the moment, we have not discussed the subordinate relationship among intents, acts and slots, which is essential to dialogue understanding.

REFERENCES

- Ankur Bapna, Gokhan Tür, Dilek Hakkani-Tur and Larry Heck. 2017. Sequential Dialogue Context Modeling for Spoken Language Understanding. *arXivpreprint*.arXiv:1705.03455.
- B.Liu and I. Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling.*arXivpreprint*.arXiv:1609.01454.
- Bordes, Y. L. Boureau, and J. Weston.2017. Learning end-to-end goal-oriented dialog. In *Proceedings of the 2017 International Conference on Learning Representations (ICLP)*.
- Yun-Nung Chen, Dilek Hakkani-Tür, GokhanTür et al. 2016. End-to-End Memory Networks with Knowledge Carryover for Multi-Turn Spoken Language Understanding. In *Proceedings of the 2016 Meeting of the International Speech Communication Association*.
- DilekHakkani-Tür, GokhanTür, AsliCelikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-Domain Joint Semantic Frame Parsing using Bi-directional RNN-LSTM. In *Proceedings of the 2016 Annual Conference of the International Speech Communication Association*.
- E. Grave, P. Bojanowski, P. Gupta, A. Joulin, T. Mikolov, et al. 2018. Learning Word Vectors for 157 Languages. In *Proceedings of the 2018 International Conference on Language Resources and Evaluation(LREC)*.

- H. Zhou, M. Huang, and X. Zhu. 2016. Context-aware natural language generation for spoken dialogue systems. In *Proceedings of the 2016 International Conference on Computational Linguistics (ICCL)*, pages 2032–2041.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint*. arXiv:1810.04805.
- P. Xu and R. Sarikaya. 2013. Convolutional neural network based triangular crf for joint intent detection and slot filling. In *Proceedings of the 2013 Automatic Speech Recognition and Understanding (ASRU)*. IEEE Workshop on. IEEE, pp. 78–83.
- Raghav Gupta, Abhinav Rastogi, and Dilek Hakkani-Tür. 2018. An Efficient Approach to Encoding Context for Spoken Language Understanding. *arXiv preprint*. arXiv:1807.00267.
- Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston and Rob Fergus. 2015. End-to-end Memory Networks. In *Proceedings of the 2015 Conference on Neural Information Processing Systems (NIPS)*.
- Shah P, Hakkani-Tür, Dilek, Tür, Gokhan, et al. 2018. Building a Conversational Agent Overnight with Dialogue Self-Play. *arXiv preprint*. arXiv:1801.04871.
- T.-H. Wen, M. Gasic, N. Mrksic, P.-H. Su, D. Vandyke, and S. Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 1711–1721, Lisbon, Portugal. <https://www.aclweb.org/anthology/D15-1199>.
- T.-H. Wen, D. Vandyke, N. Mrksic, M. Gasic, L. M. Rojas Barahona, P.-H. Su, S. Ultes, and S. Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, pages 438–449, Valencia, Spain. <https://doi.org/10.18653/v1/E17-1042>.
- Williams. 2013. Multi-domain learning and generalization in dialog state tracking. In *Proceedings of the 2013 Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*. Association for Computational Linguistics, pages 433–441. <https://www.aclweb.org/anthology/W13-4068>.
- X. Zhang and H. Wang. 2016. A joint model of intent determination and slot filling for spoken language understanding. In *Proceedings of the 2016 International Joint Conference on Artificial Intelligence (IJCAI)*.