Neural Network-based Task-oriented Dialogue System

Jiadong Zhu (jz2653) / Juanlu Yu (jy2234)

May 2018

1 Introduction

In this project, we developed a neural network based task-oriented dialogue system. The problem setting is based on WOZ 2.0 dataset [1], which is designed to help users find a restaurant based on their preference. In each round of a dialog, the system will track current *dialog state*, predict user's action (represented in "slots") and the best reply template, and finally, fill the target restaurant information in the template to produce a complete reply.

1.1 slots

Slots in each round of dialog reflect user's intention, which are made up of several requestable slots and informable slots. Each slot consists of a key and a value. For requestable slots, `R(key, boolean)` indicates that, at this moment, the user is requesting the `key` attribute of the target restaurant. For example, a state with `R(address, true)` means the user is requesting the address of a previously mentioned restaurant.

For informable slots, `I(key, value)` indicates that the user is searching for a restaurant that has the attribute `key` with value `value`. For example, a state with `I(food, japanese)` shows that the user wants to find a Japanese restaurant.

2 Model

The model we use for this project is referred to [2]. It includes following parts: user utterance encoder, state tracker, slot models, template model, knowledge base (KB), and response sentence generator. The dataflow of the model is shown in figure 1.

At each round of a dialog, the user utterance sentence is first converted into a dense vector by an encoder; then, the vector is sent into the state tracker to update it's internal state and current dialog state. With the dialog state, current slots status can be predicted with slot models for each type of slots. After that,

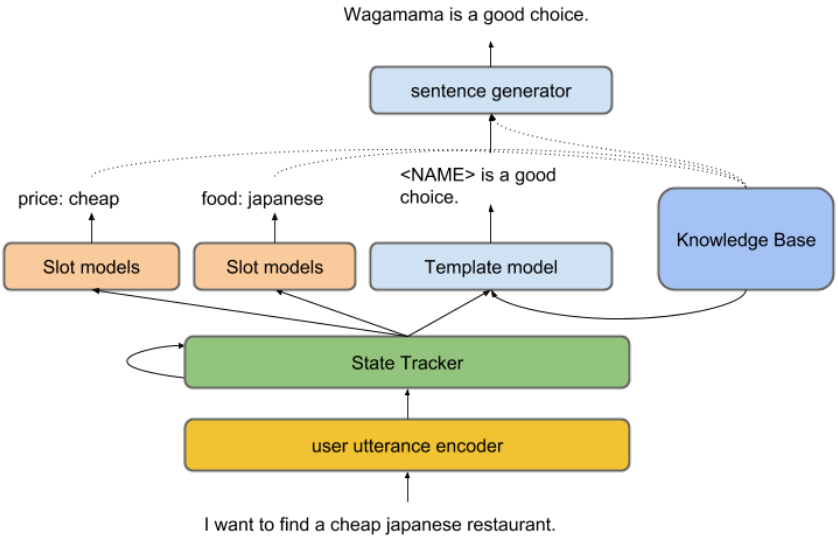

Figure 1: Model used in this work

a system response template is predicted with the information of the state tracker state and the result queried from the KB. Finally, the sentence generator will fill in the sentence template based on current context and information retrieved from the KB.

2.1 User utterance encoder

To feed the user utterance into a state tracker, it's important to encode the user utterance, a sentence, to a fixed length dense vector. In this work, the utterance is first tokenized and transformed to an array of word vectors with a word2vec model, for which, we use Paragram-SL999 [3] word vector model, since it provides better state tracking performance as [1] shows.

Then, the array of word vectors needs to be converted into a sentence vector. To achieve this, we use Bi-directional Gated Recurrent Unit Recurrent Neural Network (GRU-RNN)—each item in the word vector array is the input to the RNN and the sentence vector is the output at the last timestamp.

$$\mathbf{u} = BiRNN(\{w_1, w_2 \cdots w_n\}) \tag{1}$$

We also tried to integrate attention model[4] to the encoder, but unfortunately, it turns out that the system performance in this task does not improve.

2.2 State Tracker

The state tracker is the most important part of the whole system. As its name shows, it "tracks" the current dialog state, namely a dense vector encodes the current dialog context.

In this project, we simply use a Bi-directional GRU-RNN. It accepts user utterance vector in each round and updates its internal state.

$$h_i = BiRNN(u_i, h_{i-1}) \tag{2}$$

2.3 Slot Models

For each type of slots (food type, area preference...), we use a simple single layer dense neural network (DNN) model to predict the value of this slot.

The value v_i^t of slot t at round i can be represented by:

$$v_i^t = g(W^t h_i + b^t) \tag{3}$$

where g is the activation function, h_i is the dialog state vector at round i, W and b are trainable parameters.

2.4 Template Model

For now, we get a state tracker that can reason about the intention of a user utterance in a certain round of a dialog. Then, we need to generate the system reply.

In [5], they use a LSTM RNN based reply generator to directly generate the delexicalized reply for each round. We found that it's some kind of hard to train the model: the gradient signal from the loss is noisy due to the existing of multiple legit system replies for each user input. For example, if user asks "Can you recommend me some Chinese restaurant?", the ground truth may be a directly recommendation such as "I recommend ..." or may ask for more conditions like "we have a couple of Chinese restaurants. would you like information on those?". So it's hard to train a satisfying model based on such data setting.

Thus, instead, we choose an easier way to produce the reply. We simply map all the replies in the training set to 200 types using KMeans clustering algorithm, then use a neural network model to predict the corresponding reply type in each round.

The template model we choose is a simple 2-layer DNN. The input is a concatenation of dialog state h_i and knowledge base indicator I_i:

$$I_i = \begin{cases} \{1,0,0\}, & \textbf{if } \text{len(result)} = 0 \\ \{0,1,0\}, & \textbf{if } \text{len(result)} = 1 \\ \{0,0,1\}, & \textbf{if } \text{len(result)} > 1 \end{cases} \tag{4}$$

where `len(result)` is the length of the KB query result using the condition generated from the slots predictions. For example, if the slot prediction is `food:japanese` and `price:cheap`, and there is only 1 restaurant in the knowledge base, the indicator should be $\{0, 1, 0\}$.

Then, the template type s at round i can be represented as:

$$s_i = DNN(h_i \oplus I_i) \tag{5}$$

2.5 Loss function

The loss function we use in this model is the weighted sum of the cross-entropy loss of all slots and template type predictions.

$$L = \sum_{I}^{i}(\lambda \sum_{T}^{t} \textit{cross-entropy}(v_i^t, {v_{gt}}_i^t) + (1-\lambda)\textit{cross-entropy}(s_i, {s_{gt}}_i)) \tag{6}$$

where λ is a hyper-parameter between 0 and 1.

2.6 Sentence Generator

In order to generate the final answer, we need a template sentence and zero or one matched result from the knowledge base.

We keep a record of the last state to determine whether the current user utterance is related to the last one. This record will be compared with the current state by three types of slots: food type, area preference, and price range. If these three slots remain the same, the result from the last round will be reused to generate a new sentence; if not, we create a search string based on the current state, obtain matched results from the knowledge base, and randomly pick one result as the answer to the user's utterance.

Since not all of the 200 template types are suitable to generate the sentence, we refine the template sentences manually and map some sentence groups to another preferred group to avoid less appropriate answers. Also, when no restaurant meets the requirements, the sentence type will be set to a particular sentence group that only has "not found" type replies.

Once we get the result and the template sentence, we replace the correlated words in the template sentence with the correct information retrieved from the result to generate a new sentence. This sentence will be the reply to the user's request. For example, if we have a template and a restaurant as follows:

```
<sos> <v.NAME> is a <v.FOOD> restaurant in the <v.AREA> <s.AREA> . <eos>

{u'name': u'wagamama', u'area': u'centre', u'food': u'japanese',
u'phone': u'01223 462354', u'pricerange': u'expensive',
u'location': u'52.203,0.12375',
u'address': u'36 Saint Andrews Street',
u'type': u'restaurant', u'id': u'12638', u'postcode': u'C.B 2, 3 A.R'}
```

Then, generator will produce the following answer:

```
<sos> wagamama is a japanese restaurant in the centre area . <eos>
```

3 Evaluation

3.1 Dataset

The dataset we use in this project, WOZ 2.0[1], consists of 99 restaurants information and 1200 collected dialog instances in simulated user-assistant situations produced by real people with 600 in training set, 200 in developing or validation set and 400 in testing set. In each round of a dialog, current active slots are also given as the ground truth. More statistics are shown in table 1.

	# dialog	# round
train	600	2536
dev	200	830
test	400	1642

Table 1: WOZ 2.0 Dataset Statistics

3.2 State Tracker

For state tracker, we use the accuracy rate on all types of slots as the performance metric. The results is shown in table 2.

Model / Dataset	dev	test
baseline	0.9689	0.9612
+pretrained-word2vec	**0.9847**	**0.9786**
+attention	0.9826	0.9764

Table 2: State Tracker Evaluation Result (in accuracy rate)

For the baseline setting, we use a 300-length dialog state vector, with a randomly initialized 300-length word embedding model. In the training process, we apply RMSProp optimization algorithm, an 1e-3 learning rate with 0.9 momentum and 10-epoch early-stopping.

For "pretrained-word2vec" setting, we use a pre-trained Paragram-SL999 word embedding model with other hyper-parameter kept the same. For "attention" model, we add an additional dense layer in front of the hidden outputs of the utterance encoder and we use its output as the alternative sentence vector.

Overall, this model turns out to be pretty fit for this problem and, by inspecting the result, we could find that using a pretrained word2vec model does provide a promising performance boost. On the other side, the attention model integration failed to improve the performance as expected.

Slot	Accuracy	Precision	Recall	F-1
phone_req	0.9878	0.9809	0.9809	0.9809
area_req	0.9908	0.9246	0.7394	0.8054
area	0.9744	0.9738	0.9617	0.9675
food	0.9091	0.6447	0.5875	0.5924
pricerange_req	0.9933	0.9417	0.8194	0.8704
postcode_req	1.0000	1.0000	1.0000	1.0000
address_req	0.9860	0.9743	0.9802	0.9772
pricerange	0.9549	0.9612	0.9431	0.9515
name_req	0.9957	0.4979	0.5000	0.4989
food_req	0.9945	0.9594	0.8865	0.9197

Table 3: Evaluation Result in Detail

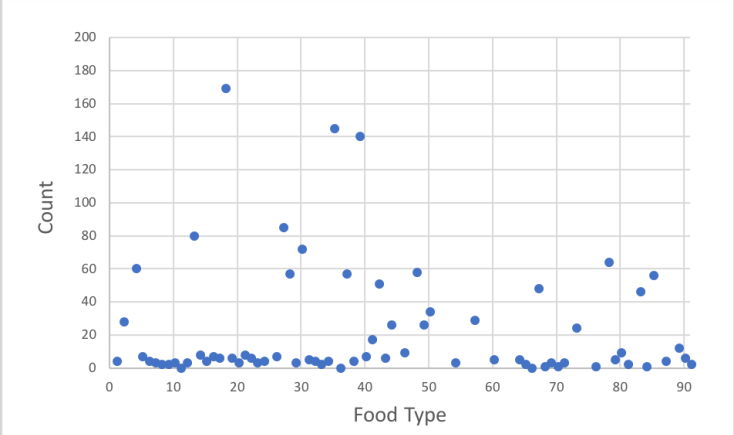

Figure 2: Food Type Count in Training Set

A more detailed evaluation result including per-slot accuracy, precision, recall and F1-score is shown in table 3, which enables us to look closer into the system performance (we use macroaveraging strategy to deal with multiclasses). Though some slot types do have a good accuracy score, there are several F1 scores revealing a significant difference. For those requestable slots, we believe that's because the "don't care" class for each slot type is pervasive in the dataset. This phenomenon is especially salient for the name request, we believe that's related to the lack of instances in which user is asking for a restaurant's name.

For food slot, we think the low F1 score is mainly attributable to the poor performance of some minority food types caused by the lack of data. That can be proved by the food type count (in training set) and confusion matrix (in testing set) presented in figure 2 and 3.

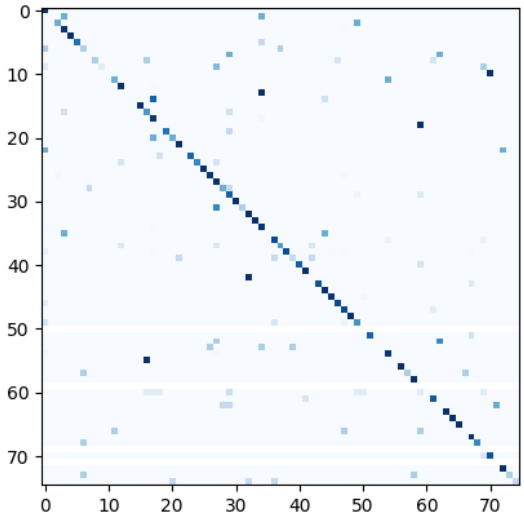

Figure 3: Confusion Matrix for Food Type Slot (testing set)

3.3 Reply Generation

For reply generation part, it's hard to find a automatic quantization method to measure its performance. In [5], they hire human evaluators to help give a "success" rate (the rate that a dialog can end successfully) and a subjective score measuring the "natualness" of the system replies, which is obviously not viable for this course project. Therefore, we just test the system by ourselves with some test cases and several system generated examples are presented below. (The lines with `<sos>` and `<eos>` are system generated replies and the others are user input.)

```
1. I'd like to have some Chinese food.
   <sos> i recommend shanghai family restaurant . <eos>
   May I know the address?
   <sos> the address is 39 Burleigh Street City Centre .
         anything else i can help you with? <eos>
   And phone number please.
   <sos> 01223 301761 . anything else i can help you with? <eos>
   No thank you.
   <sos> good bye.
         thank you for contacting the cambridge restaurant system. <eos>

2. Any cheap restaurant in the east?
   <sos> the missing sock is an cheap restaurant on the east area . <eos>
   What kind of food do they serve?
```

```
   <sos> they serve international food . <eos>

3. Are there any eritrean restaurants in town?
   <sos> sorry , there are no restaurants serving
         eritrean food available. <eos>
   How about korean food?
   <sos> little seoul is an korean restaurant in the expensive pricerange
         and the centre area , the address is 108 Regent Street City
         Centre and the phone is 01223 308681 <eos>
   Thank you so much
   <sos> you're welcome. <eos>

4. Something moderately priced in the south part of town.
   <sos> restaurant alimentum is a modern european restaurant in the
         south area and is in the moderate pricerange . the phone
         is 01223 413000 . <eos>
   I perfer italian food.
   <sos> pizza hut cherry hinton is a nice restaurant serves italian
         if you want something in the south . <eos>

5. I want to find a cheap chinese restaurant in center area
   <sos> oh wait , there are some chinese restaurants in the centre
   . the one i recommend is golden house in the cheap pricerange . <eos>
   address and phone
   <sos> the address for golden house is 12 Lensfield Road City Centre .
   is there any other information you require? <eos>
   phone please
   <sos> the phone for the lucky star is 01223 244277 . can i help you
   with anything else today? <eos>
   that's all. thanks
   <sos> goodbye <eos>
```

In our experience, it's hard to say this part's performance is satisfying. We think that is due to the noisy ground truth as mentioned in section 2.4 and the lack of data is also another possible reason. A probably reasonable approach is to create some artificial reply templates and choose the type directly based on the slot prediction as in [6]. Since the state tracker we have does have a good accuracy rate.

4 Conclusion

In this project, we developed a neural network based task-oriented dialogue system. It is capable of parsing user intentions from their utterance input and generating system response based on a tracked dialog state. In our evaluation, it shows that our system gives a reasonable performance but there is still a large space to improve.

References

[1] Nikola Mrksic, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve J. Young. Neural belief tracker: Data-driven dialogue state tracking. *CoRR*, abs/1606.03777, 2016.

[2] Bing Liu and Ian Lane. An end-to-end trainable neural network model with belief tracking for task-oriented dialog. *CoRR*, abs/1708.05956, 2017.

[3] John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Towards universal paraphrastic sentence embeddings. *CoRR*, abs/1511.08198, 2015.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[5] Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina Maria Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, and Steve J. Young. A network-based end-to-end trainable task-oriented dialogue system. *CoRR*, abs/1604.04562, 2016.

[6] Xiujun Li, Yun-Nung Chen, Lihong Li, and Jianfeng Gao. End-to-end task-completion neural dialogue systems. *CoRR*, abs/1703.01008, 2017.