Encoder-Memory-Decoder Model for Long Conversation Dialogue System

Changliang Li, Xiuying Wang, Jian Cheng, Bo Xu
Institute of Automation Chinese Academy of Sciences
95 Zhongguancun East Road, 100190, BEIJING, CHINA
{changliang.li,xiuying.wang,jcheng,xubo}@ia.ac.cn

Abstract. One long-term goal in artificial intelligence field is to build an intelligent dialogue agent. Recently with the development of deep learning, the popular dialogue system is built on the encoder-decoder framework for sequence-to-sequence learning just like Neural Machine Translation. However, this approach can only handle single turn dialogue without consistency, due to its lack of ability to acquire the dialogue history information. It’s still challenging to build a dialogue system that works reasonably well for long conversations (multiple turns). In this paper, we propose an Encoder-Memory-Decoder model to build long conversations dialogue system in neural generative way. It can be viewed as an end-to-end neural network model equipped with memory ability to memorize the dialogue history information for generative dialogue response. More specifically, the proposed model requires few hand-crafted rules and generates more flexible responses. Empirical study shows the proposed model can effectively deal with long conversations, and can generate right and natural response coherently. This model gives a new perspective for building long conversation dialogue system.

Keywords: Encoder-Memory-Decoder; Long Conversation Dialogue

1 Introduction

Dialogue systems, also known as interactive conversational agents, virtual agents or sometimes chatterbots, are useful in a wide range of applications ranging from technical support services to language learning tools and entertainment [1,2]. There is a rich history of dialogue system research. During the past decades, the research of dialogue system has witnessed much progress. Generally, Dialogue systems can be divided into two kinds of models: Retrieval-based and Generative models.

Retrieval-based models use a repository of predefined responses and some kind of heuristic to pick an appropriate response based on the input and context. The heuristic could be as simple as a rule-based expression match, or as complex as an ensemble of Machine Learning classifiers [3,4]. These systems don’t generate any new text; they just pick a response from a fixed set. Due to the repository of handcrafted responses, retrieval-based methods don’t make grammatical mistakes. However, they may be

© Springer-Verlag Berlin Heidelberg 2011
unable to handle unseen cases for which no appropriate predefined response exists. For the same reasons, these models can’t refer back to contextual entity information like names mentioned earlier in the conversation.

Generative models don’t rely on pre-defined responses. They generate new responses from scratch. Deep Learning architectures like Sequence-to-Sequence are uniquely suited for generating text and researchers are hoping to make rapid progress in this area. Generative models are typically based on Encoder-Decoder model. It is introduced from Machine Translation techniques, but instead of translating from one language to another, it “translate” from an input to an response [5]. Generative models are “smarter”. They can refer back to entities in the input and give the impression that is talking to a human. However, End-to-End model can only handle Short-Text Conversations, also called single turn dialogue without taking dialogue history information into account.

It’s still at the early stages of building generative models that work reasonably well, especially for long conversations. In long dialogues people need to go through multiple turns and keep track of what has been said as well as what information has been exchanged. The most common approach is to embed the conversation into a vector, but doing that with long conversations is challenging. Work of [6] goes into that direction. One may also need to incorporate other kinds of contextual data such as date/time, location, or information about a user. To our best knowledge, there are no dialogue systems that can handle long conversation well.

Our work is partly inspired by recent progress in memory networks, which has been proved its efficiency in QA task [7,8]. Memory network can memorize the relevant knowledge, for example previous turns dialogue.

In this paper, we extend and improve Encoder-Decoder model via introducing memory module. We propose Encoder-Memory-Decoder model. This model takes advantage of both memory networks, which can memorize dialogue history information, and encoder-decoder model, which can response in a neural generative way. Our model can be viewed as a cognitive system, which has to carry out natural language understanding, reasoning, memorizing and natural language generation.

We conduct experiment on long conversation corpus of different domains. And the result shows that our model can handle long conversation dialogue well and generate smooth response. And in grammar accurate, it is competitive with retrieval-based models in the literature. This paper gives a novel way to build long conversation dialogue system.

2 Related Work

Our work is inspired by recent work on sequence-to-sequence framework [9,10] and memory networks [7,11].
2.1 LSTM Networks

One important property of tasks based on natural languages, is that we deal with variable-length input $X = (x_1, x_2, ..., x_T)$ and output $Y = (y_1, y_2, ..., y_{T'})$. $T$ and $T'$ are not fixed.

To deal with these types of variable-length input and output, we need to use a recurrent neural network (RNN). The main idea behind RNNs is to compress a sequence of input symbols into a fixed-dimensional vector by using recursion.

Long Short Term Memory networks (referred as LSTMs) are a special kind of RNN, capable of learning long-term dependencies. They were introduced by [12], and were refined and popularized by many people in following work. They work tremendously well on a large variety of problems.

LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their feature, and it has been widely used in machine learning and artificial intelligence filed.

2.2 Sequence-to-Sequence Framework

A basic sequence-to-sequence model, as introduced in [13] consists of two recurrent neural networks (RNNs): an encoder that processes the input and a decoder that generates the output. Figure 1 shows this basic architecture.

![Sequence-to-Sequence Framework Diagram](image)

Figure 1 represents Sequence-to-Sequence Framework, and each box represents most commonly a GRU cell or an LSTM cell. Encoder and decoder can share weights or, as is more common, use a different set of parameters. Multi-layer cells have been successfully used in sequence-to-sequence models too, e.g. for translation [10]. This framework has also been used in natural language dialogue [6,14-17] where the end-to-end neural dialogue model is trained on a large amount of conversation data. Although promising, neural dialogue models still have problems and limitations, e.g., the lack of ability to handle long conversations.

In the basic model depicted above, every input has to be encoded into a fixed-size state vector, as that is the only thing passed to the decoder. To allow the decoder more direct access to the input, an attention mechanism was introduced in [9]. It allows the decoder to peek into the input at every decoding step. There are many different se-
quence-to-sequence models. Each of these models can use different RNN cells, but all of them accept encoder inputs and decoder inputs.

2.3 Memory Networks

A memory network combines learning strategies from the machine learning literature with a memory component that can be read and written to [11]. Figure 2 shows a typical memory networks.

![Fig. 2. Memory Networks](image)

The model is trained to learn how to operate effectively with the memory component. And memory networks have been successfully applied to Question and Answer task. The high-level view of a memory network is as follows:

- There is a memory, \( m \), an indexed array of objects (e.g. vectors or arrays of strings).
- An input feature map \( I \), which converts the incoming input to the internal feature representation
- A generalization component \( G \) which updates old memories given the new input. There is an opportunity for the network to compress and generalize its memories at this stage for some intended future use.
- An output feature map \( O \), which produces a new output in the feature representation space given the new input and the current memory state.
- A response component \( R \) which converts the output into the response format desired – for example, a textual response or an action.

\( I, G, O \) and \( R \) can all potentially be learned components and make use of any ideas from the existing machine learning literature.

The most basic version of memory networks works as follows. \( I \) is given a sentence at a time, and \( G \) simply stores the sentence in the next available memory slot (i.e., we assume there are more memory slots than sentences). All of the hard work is done in the \( O \) and \( R \) components. \( O \) is used to find up to \( k \) supporting memories.
3 Encoder-Memory-Decoder Model

In this section, we describe our Encoder-Memory-Decoder model in details. Our model takes advantage of both memory networks and encoder-decoder model. Figure 3 shows our Encoder-Memory-Decoder model.

Our model extends and improves Encoder-Decoder framework by introducing memory module. We take human input dialogue \( D \) as a sequence of \( N \) tokens \( I_n = (I_n^1, I_n^2, ..., I_n^N) \). Each element \( I_n^i \) represents one token. In our work tokens represent both words and speech acts, for example pause and end of turn tokens. We take the dialogue history as knowledge base represented as \( H_{i_s} = (h_i^{s1}, h_i^{s2}, ..., h_i^{sM}) \). Each unit \( h_i^{s1} \) contains \( M_i \) tokens. The output, also called machine response, is represented as a sequence of \( K \) tokens \( R = (R_1, R_2, ..., R_K) \). After generating the machine response, the model’s knowledge base module adds the response as the next unit \( h_{i_s}^{M+1} \). Through this process of iteration, the model is equipped with ability to acquire dialogue history information.

In the Encoder–Decoder framework, an encoder reads the input sentence, a sequence of vectors \( I = (I_1, I_2, ..., I_N) \), into a fixed-length vector as calculated by Eq.(1) and Eq.(2).

\[
\begin{align*}
h_t &= f(I_t + h_{t-1}) \quad (1) \\
c_{in} &= q((h_1, ..., h_T)) \quad (2)
\end{align*}
\]

\( h_t \in R^n \) is a hidden state at time \( t \), and \( c_{in} \) is a vector generated from the sequence of the hidden states. \( f \) and \( q \) are nonlinear functions. In our work, \( f \) is set as
the sophisticated long short-term memory (LSTM) unit. Because LSTM is specially designed for its long term memory: it can store information over extended time steps without too much decay. Figure 4 gives the graphical model of the encoder.

![Graphical Model of Encoder](image)

**Fig. 4.** Encoder

We essentially use the final hidden state $h_T$ as the global representation of the sentence as shown in Eq.(3). The same strategy has been taken in [10,13] for building the intermediate representation for machine translation.

$$q([h_3, ..., h_T]) = h_T$$  \hspace{1cm} (3)

The history turns of dialogue, referred as knowledge base, are fed into two tunnel encoders: representation encoder and memory encoder. The representation encoder is applied to represent dialogue history information as knowledge vector by Eq.(4) and Eq.(5).

$$h_{kb_t} = f(h_{is_t} + h_{kb_{t-1}})$$  \hspace{1cm} (4)

$$c_{kb} = q([h_{kb_1}, ..., h_{kb_T}])$$  \hspace{1cm} (5)

Where $h_{kb_{t-1}}$ is a hidden state at time $t$, and $c_{kb}$ is a vector generated from the sequence of the hidden states.

And memory encoder is applied to represent dialogue history information as memory slots by Eq.(6) and Eq.(7).

$$h_{kb_m_t} = f(h_{is_t} + h_{kb_{m_{t-1}}})$$  \hspace{1cm} (6)

$$c_{kb_m} = q([h_{kb_m_1}, ..., h_{kb_m_T}])$$  \hspace{1cm} (7)

Where $h_{kb_{m_{t-1}}}$ is a hidden state at time $t$, and $c_{kb_m}$ is a vector of memory slot generated from the sequence of the hidden states.

The input representation and memory slots are matched to obtain hidden representation $HidVec$ as shown in Eq.(8).

$$HidVec = Match(c_{in}, c_{kb_m})$$  \hspace{1cm} (8)
*Match* is a function that implements matrix dot computation.

And then the hidden representation *HidVec* is fed to merge with knowledge vector to get a fixed-length output vector *OutVec* as shown in Eq.(9).

\[
\text{OutVec} = \text{Merge}(\text{HidVec}, c_{kb})
\]  

(9)

*Merge* is a function that contacts the two vectors into a fixed-length vector.

After the merger process, we get the output representation *OutVec*. Based on *OutVec*, the decoder starts producing a response sentence. Figure 5 gives the graphical model of the decoder.

The decoder of the proposed model is another RNN which is trained to generate the output sequence by predicting the next symbol \(y_t\) given the hidden state \(s_t\). Both \(y_t\) and \(s_t\) are conditioned on \(y_{t-1}\) and on *OutVec*. Hence, the generation probability of the \(t\)-th word is calculated by Eq.(10).

\[
\begin{align*}
\text{p}(y_t | y_{t-1}, \ldots, y_1, X) & = g(y_{t-1}, s_t, \text{OutVec}) \\
\text{g} & \text{ is a softmax activation function, and } s_t \text{ is the hidden state of decoder at time } t \text{ calculated by } \text{Eq.(11).} \\
s_t & = f(y_{t-1}, s_{t-1}, \text{OutVec})
\end{align*}
\]

(10)  

(11)

The decoder keeps generating words until a special end of sentence token (for example <EOS>) is produced.

4 Experiment

We conduct experiments on three different domains dataset and compare our Encoder-Memory-Decoder model to retrieval-based model just like in [6].
4.1 Dataset

Firstly, we describe our dataset. To construct dataset, we first collect human-human dialogues in different domains, such as weather query, bank service and booking restaurant. And then we filter the raw data according to some simple rules, for example deleting long sentences which conclude more than 20 words. We followed the examples in the corpus and manually build a set of dialogue. Table 1 shows some dialogue samples in dataset. M represent machine response and U represents user input.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Dialogue Example</th>
<th>Translation</th>
</tr>
</thead>
</table>
As a result, we get the dataset of three domains as shown in Table 2.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Number of samples (Train)</th>
<th>Number of samples (Test)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restaurant</td>
<td>1393</td>
<td>689</td>
</tr>
<tr>
<td>Weather</td>
<td>2455</td>
<td>792</td>
</tr>
<tr>
<td>Bank</td>
<td>2834</td>
<td>815</td>
</tr>
</tbody>
</table>

4.2 Evaluation Metrics

Evaluation metrics of a dialogue system is still an open problem [18,19,20]. There is no well-established method for automatic evaluation, and human-based evaluation is expensive. Word error-rate is a well-established performance metric for probabilistic language models [6]. Similarly, in our work we employ word accurate-rate as one evaluation metric. This is defined as the number of words in the dataset the model has predicted correctly divided by the total number of words in the dataset.

Similar to word accurate-rate, we propose response semantic accurate-rate. This is defined the number of predicted responses of same semantic meaning with target response divided by the total number of response in the dataset. For example, in weather query field, user said “I want to know the weather”, and the target response is “which city do you want to query?”. If the model predicts the response as “where?”, though there is no same words between predicted sentences and target sentences, it will be viewed as a right response due to both sentences are of same meaning. We define two sentences with same semantic meaning as the semantic similarity score between two sentences is higher than a predefined threshold.

Ultimately, we care about the fluency of response, syntactically and semantically coherent dialogues. This is difficult to measure. We measure it by human judge.

4.3 Implementation details

We use ICTCLAS to split the sentences into sequences of words. Since the word distributions on user inputs and machine responses are different, we build two different vocabularies. All the out-of-vocabulary words are replaced by a special token “UNK”. The dimensions of the hidden states of encoder/decoder and RNN/RNN-lstm are all set to 128. And the dimension of the word-embedding is set to 64. The word-embeddings are initialized by randomly sampling from a uniform distribution. All models are trained on a K20 GPU with the mini-batch size of 12.
4.4 Results

Following work in [6,14] we compare our model to retrieval-based methods, the feature of which is that all responses are retrieved from the train set. Like work [6], we take standard-RNN and RNN-lstm networks as baselines. Table 3 shows some result examples of different models. T means the turn of dialogue.

Table 3. Result examples

<table>
<thead>
<tr>
<th>T</th>
<th>User input</th>
<th>Target response</th>
<th>EMD</th>
<th>RNN</th>
<th>RNN-lstm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>您好, 我要定位子。 (hello, I need reserving a table)</td>
<td>一共多少人? (how many people)</td>
<td>您几位? (how many people)</td>
<td>你想要什么时候的座位啊? (What time do you need to reserve?)</td>
<td>叫什么名字? (what's your name?)</td>
</tr>
<tr>
<td>2</td>
<td>有6个人。 (six)</td>
<td>订什么时间的啊? (What time do you need to reserve?)</td>
<td>您什么时候要啊? (What time will you come?)</td>
<td>那你告诉我一下你名字。 (can you tell me your name.)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>请帮我预订一下这周六上午十点的座位。 (Please reserve me at 10:00 this Saturday morning.)</td>
<td>没问题，您怎么称呼? (No problem, what’s your name?)</td>
<td>可以，这个你称呼名字。 (ok, what’s your name.)</td>
<td>您几点来就餐啊? (What time will you come?)</td>
<td>没问题，您告诉我一下您名字。 (no problem, can you tell me your name.)</td>
</tr>
<tr>
<td>4</td>
<td>名是我涛 (my name is Tao Zhou)</td>
<td>好的，周六上午十点的6人桌。 (Ok, Mr.Zhou, we have reserved for you a 6-person table at 10:00 on Saturday morning.)</td>
<td>好的，周涛，为您预订了这周六上午十点的1人桌。 (Ok, Mr.Zhou, we have reserved for you a 1-person table at 10:00 on Saturday morning.)</td>
<td>好的，周涛，为您预订了这周六上午七点半的1人桌。 (Ok, Mr.Zhou, we have reserved for you a 1-person table at 7:30 this morning.)</td>
<td>好的，周涛，已经为您预订了这周六上午十点的一人桌。 (Ok, Mr.Zhou, we have reserved for you a 1-person table at 10:00 on Saturday morning.)</td>
</tr>
<tr>
<td>5</td>
<td>非常好。谢谢。 (good, thank you)</td>
<td>不客气，期待您的光临。 (You're welcome, We look forward to your visit)</td>
<td>不用谢，为您服务是我们的荣幸。 (You're welcome, It is our pleasure to serve you.)</td>
<td>好的，已经为您预订好，谢谢您的来电。 (OK. We've made the reservation for you. Thanks for calling.)</td>
<td>好的，已经为您预订好，谢谢您的来电。 (OK. We’ve made the reservation for you. Thanks for calling.)</td>
</tr>
</tbody>
</table>
Table 4 shows the word accurate-rate and semantic accurate-rate of the models on test sets.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Model</th>
<th>Word (%)</th>
<th>Semantic (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weather</td>
<td>RNN</td>
<td>29.28</td>
<td>42.03</td>
</tr>
<tr>
<td></td>
<td>RNN-lstm</td>
<td>29.10</td>
<td>41.98</td>
</tr>
<tr>
<td></td>
<td>EMD</td>
<td>68.44</td>
<td>69.48</td>
</tr>
<tr>
<td>Restaurant</td>
<td>RNN</td>
<td>20.92</td>
<td>33.10</td>
</tr>
<tr>
<td></td>
<td>RNN-lstm</td>
<td>19.78</td>
<td>32.20</td>
</tr>
<tr>
<td></td>
<td>EMD</td>
<td>40.25</td>
<td>48.22</td>
</tr>
<tr>
<td>Bank service</td>
<td>RNN</td>
<td>22.92</td>
<td>36.17</td>
</tr>
<tr>
<td></td>
<td>RNN-lstm</td>
<td>22.00</td>
<td>35.51</td>
</tr>
<tr>
<td></td>
<td>EMD</td>
<td>49.50</td>
<td>52.84</td>
</tr>
</tbody>
</table>

We can see that our Encoder-Memory-Decoder model, referred as EMD, performs much better than RNN and RNN-lstm. RNN and RNN-lstm performs no big difference on this task. The result can be explained that retrieval based method lacks the ability of generalization, but our model does. For example, in test set of booking restaurant domain, the target response “那 你 告诉 我 一下 名字”(can you tell me your name ?) isn’t seen in train set. Then it cannot be retrieved and gives a wrong answer. However, in generative way, the response is composed of words. Though some responses are not seen in train set, each word in these responses is seen in train set. So our model can solve the unseen problem to a large extent.

We make some empirical comparisons and find that no significant differences between retrieval based method and EMD in terms of the fluency of response. For example, in the result example shown in Table 3, the response of the third turn, “可以，这个你称呼名字。” though there is some noise words such as “这个”(this), but human can easily understand the meaning of the sentence, which means asking about the user’s name. In general, our EMD model yields responses that human can understand its meaning in most of the time.

5 Conclusion

In this paper we introduced a novel model called Encoder-Memory-Decoder to build long conversation dialogue system. The model is built on the encoder-decoder framework for sequence-to-sequence learning, while introducing memory module as to memorize dialogue history information. It can be viewed as an end-to-end neural network model equipped with memory ability for generative dialogue response. Empirical studies show the proposed model is capable of generating natural and right response to the user input based on the dialogue history information. Future work will
continue working on building dialogue system in generative way and explore multi-modal inputs.

Reference