Context-aware Natural Language Generation for Spoken Dialogue Systems

Hao Zhou, Minlie Huang, Xiaoyan Zhu
State Key Laboratory of Intelligent Technology and Systems,
National Laboratory for Information Science and Technology,
Dept. of Computer Science and Technology, Tsinghua University, Beijing 100084, PR China
tuxchow@gmail.com, aihuang@tsinghua.edu.cn, zxy-dcs@tsinghua.edu.cn

Abstract

Natural language generation (NLG) is an important component of question answering(QA) systems which has a significant impact on system quality. Most tranditional QA systems based on templates or rules tend to generate rigid and stylised responses without the natural variation of human language. Furthermore, such methods need an amount of work to generate the templates or rules. To address this problem, we propose a Context-Aware LSTM model for NLG. The model is completely driven by data without manual designed templates or rules. In addition, the context information, including the question to be answered, semantic values to be addressed in the response, and the dialogue act type during interaction, are well approached in the neural network model, which enables the model to produce variant and informative responses. The quantitative evaluation and human evaluation show that CA-LSTM obtains state-of-the-art performance.

1 Introduction

Natural language generation (NLG), the task of generating natural language from a knowledge base or a logical form representation, is an important component of dialogue or question answering system. NLG can be treated as a single-turn dialogue generation. Traditional approaches to NLG problem are mostly rule-based or template-based (Bateman and Henschel, 1999; Busemann and Horacek, 2002). However, these methods tend to generate rigid and stylised language without the natural variation of human language. In addition, they need a heavy workload to design the templates or rules.

Recently due to the growth of artificial neural networks and the increase of labeled data available on the Internet, data-driven approaches are developed to attack the NLG problem (Ritter et al., 2011; Shang et al., 2015). Shang et al. (2015) and Serban et al. (2015) apply the RNN-based general encoder-decoder framework to the open-domain dialogue response generation task. Although their model can generate the relevant and variant responses according to the input text in a statistical manner, the quality and content of responses depend on the quality and quantum of the training corpus. Wen et al. (2015) propose a task-oriented NLG model that can generate the responses providing the correct answers given the dialogue act (for instance, confirm or request some information), including the answer information. However the context information, such as the input question and dialogue act, is ignored. Yin et al. (2016) propose a neural network model that can generate answers to simple factoid questions based on a knowledge base. But a large error rate is observed due to the complex architecture introduced.

In this paper, we deal with the NLG problem in this setting: given a question, the corresponding dialogue act, and the semantic slots to be addressed in the response, how to generate a natural language response in a dialogue. We present a statistical task-oriented NLG model based on a Context-Aware Long Short-term Memory network (CA-LSTM), which adopts the general encoder-decoder framework to incorporate the question information, semantic slot values, and dialogue act type to generate correct answers. The major departures from prior work lie in:

- We design a context-aware generation framework for NLG. The framework incorporates the embedding of question and dialogue act type, and semantic values to be satisfied in the generated

This work is licenced under a Creative Commons Attribution 4.0 International License. License details: http://creativecommons.org/licenses/by/4.0/

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2032–2041, Osaka, Japan, December 11-17 2016.

response.

- We propose an attention mechanism that attends the key information in question conditioned on the current decoding state of the decoder. Thus the question embedding is dynamically changed over the decoding states.

- We propose to encode dialogue act type embedding to enable the model to generate variant answers in response to different act types. In this way, the response quality is substantially improved for particular act types.

2 Related Work

Traditional NLG methods divide the task into three phases: text planning, sentence planning, and surface realization (Walker et al., 2002), which can be solved by rules or templates in traditional approach (Mirkovic and Cavedon, 2011). The quality of language generated by rule-based or template-based methods mostly depends on the handcrafted rules or templates. Even if they can ensure the adequacy, fluency and readability of the language, the need of writing and maintaining the rules or templates is a large burden to these methods. Furthermore, the rule-based or template-based methods tend to generate rigid and nonvariant responses. To address the problem, Oh and Rudnicky(2000) propose a statistical approach which can learn from data to generate variant language using a class-based n-gram language model. However, due to the limits of the model and the lack of semantically-annotated corpora, the statistical approach cannot ensure the adequacy and readability of the generated language. Thus the statistical approach has not yet been employed widely compared to the rule-based or template-based methods.

Recently due to the maturity of artificial neural networks and the rich annotated data available on the Internet, data-driven statistical approaches are developed to address the NLG problem. These methods can be divided into two categories according to the corpus type, one is open-domain chat-based NLG, the other is task-oriented NLG for solving specific tasks.

2.1 Open-domain Chat-based NLG

Chat-based NLG aims to generate relevant and coherent responses in either single-turn or multi-turn dialogues. Shang et al.(2015) propose a Neural Responding Machine (NRM), a neural network-based chatbot NLG for Short-Text Conversation, which is trained on a large amount of one-round conversation data collected from a microblogging service. NRM takes the general encoder-decoder framework: NRM encodes the input text to the latent representation and then decodes the encoded information to generate responses. Rather than the traditional NLG task which includes text planning, sentence planning, and surface realization phases, NRM formalizes the NLG task as a general decoding process based on the encoded latent representation of the input text. Serban et al. (2015) propose a hierarchical recurrent encoder-decoder neural network to the open domain dialogue. In addition to the input text, the hierarchical model encodes the context information to generate the response.

Data-driven statistical approaches have also been studied for the text planning phase. In the text planning phase, NLG chooses the proper information of every sentence to be presented to users. The generation models mentioned above are trained by predicting the system response in a given conversational context using the maximum-likelihood estimation (MLE) objective so that they tend to generate nonsense responses such as "I dont know". To address this problem, Li et al. (2016) applies deep reinforcement learning to model long-term reward in chatbot dialogue which can plan the information in the response and avoid generating the nonsense responses in the dialogue.

2.2 Task-oriented NLG for Specific Domain

The statistical methods mentioned above are designed for open-domain chatbots, which emphasize on generating relevant and fluent responses according to the input text. While these methods are not suitable for task-solving scenarios (for instance, dialogue systems for restaurant and hotel reservation), which aims at providing correct answers to the input questions, because the responses of these methods are generated from the training data, which can not contain correct answers to any questions. Wen et al.

(2015) propose a statistical NLG based on a semantically controlled Long Short-term Memory (SC-LSTM) recurrent network which partially solves this problem. In the sentence planning phase, SC-LSTM generates the answers containing the slot tokens according to the dialogue act and slot-value pairs; in the surface realization phase, SC-LSTM replaces the slot tokens with the correct answers to the input questions. However, SC-LSTM generates responses with given answers information (the dialogue act and slot-value pairs) regardless of the input questions and generated answers. A Neural Generative Question Answering (GENQA), which can generate answers to simple factoid questions based on a knowledge base, addresses the task-oriented NLG problem further (Yin et al., 2016). However after unifying understanding of question, generation of answer, and retrieval of relevant facts in a knowledge-base into an end-to-end framework, GENQA introduces more errors in answers.

3 The Context-Aware LSTM Model (CA-LSTM)

3.1 Overview

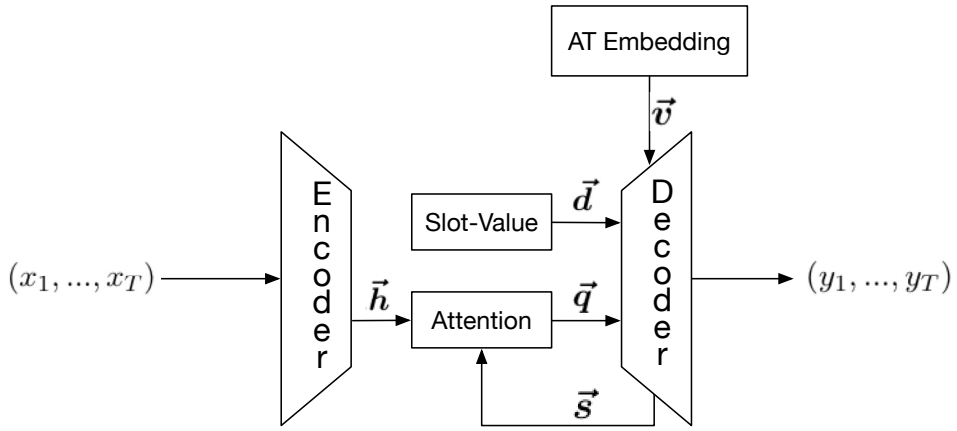

Figure 1: The general framework and dataflow of CA-LSTM. $\vec{h}$ is the hidden representation of the input question $(x_1, ..., x_T)$, $\vec{q}$ is the question vector generated by attention on $\vec{h}$ and the state of decoder $\vec{s}$, $(y_1, ..., y_T)$ is the output answer decoded with the question vector $\vec{q}$, the slot-value vector $\vec{d}$ and the dialogue act type (AT) embedding $\vec{v}$.

The Context-Aware LSTM (CA-LSTM) is built on the general encoder-decoder framework for sequence-to-sequence learning (Sutskever et al., 2014), as shown in Figure 1. Let $\boldsymbol{X} = (x_1, ..., x_T)$ and $\boldsymbol{Y} = (y_1, ..., y_T)$ denote the input question and output response respectively. The encoder converts the question sequence $\boldsymbol{X}$ to the hidden representation of the question sequence $\boldsymbol{h} = (h_1, ..., h_T)$. Then the hidden vector $\boldsymbol{h}$ is converted to the high dimensional question vector $\boldsymbol{q}$ after being attended with the current state of decoder. Finally, with the input of the question vector $\boldsymbol{q}$, the slot-value vector $\boldsymbol{d}$ and the embedding vector $\boldsymbol{v}$ of dialogue act type, the decoder generates the answer sequence $\boldsymbol{Y}$. Specifically, the slot-value vector $\boldsymbol{d}$[1] is a one-hot representation of the slot-value pairs which can regularize the generation process and ensure the information adequacy of answers as suggested by (Wen et al., 2015). By providing to the decoder the context information, including question vector and the act type embedding vector, CA-LSTM can generate context-aware responses which are related to the input question and dialogue acts.

The encoder and decoder are both based on Long Short-term Memory for its ability of modeling sequences of arbitrary lengths (Hochreiter and Schmidhuber, 1997). The procedure of generating variant responses has two components: the forward generator and the backward reranker, which share the same CA-LSTM network structure but with different parameters. To make use of the context information, the generator handles sequences in the forward direction and the reranker in the backward direction. The CA-LSTM network is introduced in Section 3.1 3.5 and the reranker is discussed in Section 3.6.

[1]The slot values are a set of values that must be satisfied in the response to be a correct answer, such as the price and type of requested restaurants.

3.2 Context-aware Decoding

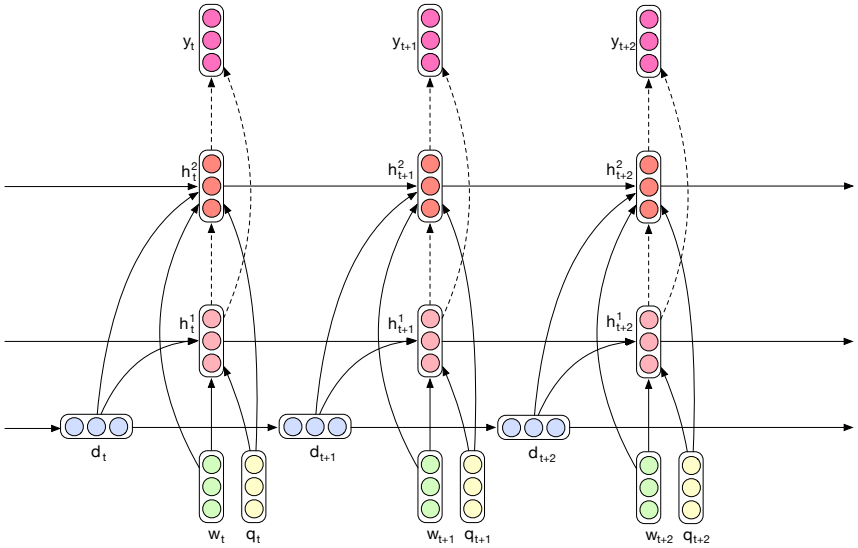

Figure 2: The graphical model of the decoder. w_t is the word embedding of the input token, q_t and d_t is the question vector and slot-value vector respectively, h_t^1 and h_t^2 are the hidden layer representations, and y_t is the probability distribution of the next token.

The decoder is based on the RNN language model (RNNLM), which takes the word embedding w_t of a token as the input at every time step t and outputs the probability distribution y_t of the next token w_{t+1} conditioned on the hidden layer representations h_t^1 and h_t^2. In addition to using the traditional LSTM cell in the RNNLM, we add the question vector $\boldsymbol{q}$ and the slot-value vector $\boldsymbol{d}$ in the network, as shown in Figure 2.

Deep neural networks can improve the network performance by learning multiple levels of feature abstraction. In order to learn more rich features, we adopt a two-layer Stacked-LSTM in the model, which has been proved effective in speech recognition (Graves et al., 2013) and other tasks. We apply the dropout operation on the dashed connections to prevent co-adaptation and overfitting, as shown in Figure 2. The Stacked-LSTM network is defined as follows,

$$i_t^n = \sigma(W_{wi}^n w_t + W_{hi}^n h_{t-1}^n + W_{hhi}^n h_t^{n-1} + W_{qi}^n q_t + b_i^n) \quad (1)$$

$$f_t^n = \sigma(W_{wf}^n w_t + W_{hf}^n h_{t-1}^n + W_{hhf}^n h_t^{n-1} + W_{qf}^n q_t + b_f^n) \quad (2)$$

$$o_t^n = \sigma(W_{wo}^n w_t + W_{ho}^n h_{t-1}^n + W_{hho}^n h_t^{n-1} + W_{qo}^n q_t + b_o^n) \quad (3)$$

$$\hat{c}_t^n = tanh(W_{wc}^n w_t + W_{hc}^n h_{t-1}^n + W_{hhc}^n h_t^{n-1} + W_{qc}^n q_t + b_c^n) \quad (4)$$

$$c_t^n = f_t^n \odot c_{t-1}^n + i_t^n \odot \hat{c}_t^n + tanh(W_{dc}^n d_t) \quad (5)$$

$$h_t^n = o_t^n \odot tanh(c_t^n) \quad (6)$$

where n and t denote the n^{th} layer in space and the t step in time respectively, σ is the sigmoid function, i_t^n, f_t^n, $o_t^n \in [0,1]^n$ are respectively the input gate, forget gate and output gate , $\hat{c}_t^n$ is the candidate cell value and c_t^n is the true cell value, and h_t^n is the hidden layer vector at time step t and layer n, all of which have the same dimension as the hidden layer vector. q_t is the question embedding vector and d_t is the slot-value vectors which will be described soon later.

Recalling that the task-oriented NLG aims at providing correct answers in responses, we thus use the slot-value vector $\boldsymbol{d}$ as the sentence planning cell to regularize the generation process and ensure the information adequacy of responses. The $\boldsymbol{d}$ vector stores a set of slot values that must be satisfied for the generated response to be qualified as a correct answer. The sentence planning cell is defined by the following equations,

$$r_t = \sigma(W_{wr} w_t + \sum_n \alpha_n W_{hr}^n h_{t-1}^n) \quad (7)$$

$$d_t = r_t \odot d_{t-1} \tag{8}$$

where $r_t \in [0,1]^d$ is the reading gate, d_0 is the initial one-hot slot-value vector by setting the corresponding bit to 1 and d_t is the slot-value vector at time step t which updates at every time step with the reading gate r_t. d_t influences the cell value c_t of the traditional LSTM cell by adding a nonlinear transformation $tanh(W_{dc}d_t)$ as shown in Equation 5.

Finally, the output probability distribution y_t of the next token w_{t+1} is computed as follows, and the next token w_{t+1} is generated by sampling from the distribution y_t.

$$w_{t+1} \sim y_t = P(w_{t+1} \mid w_t, w_{t-1}, , w_0, q_t, d_t) = softmax(\sum_n W_{ho}^n h_t^n) \tag{9}$$

3.3 Attention-based Encoding

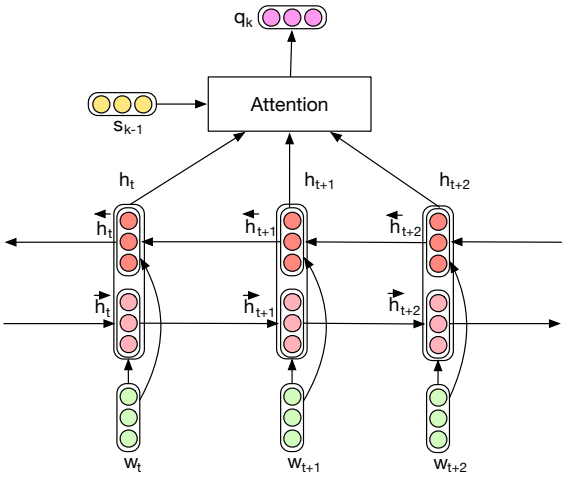

Figure 3: The graphical model of the encoder. h_t is the hidden layer representation of the input question at time step t for encoder, s_{k-1} is the state of the decoder at time step $k-1$ for decoder, q_k is the question vector at time step k for decoder generated by attention of $\boldsymbol{h}$ and the state of decoder s_{k-1}.

The encoder is built on Long Short-term Memory network. To model the question sequence with both of the preceeding and following contexts, we apply a Bidirectional-LSTM architecture (Graves et al., 2005) to the encoder as shown in Figure 3. The $\overrightarrow{h}_t$ and $\overleftarrow{h}_t$ are the two directional hidden representations which are computed by iterating the forward layer from $t = 1$ to T and the backward layer from $t = T$ to 1 respectively. By concatenating the $\overrightarrow{h}_t$ and $\overleftarrow{h}_t$, we obtain the hidden representation of the question sequence $h_t = [\overrightarrow{h}_t ; \overleftarrow{h}_t]$.

Although we can feed the hidden layer representation h_T at time step T to the decoder as the question vector q_k, this method has its shortcomings: the question is processed at one time instead of being dynamically attended as the decoder proceeds. To address this problem, the attention mechanism (Bahdanau et al., 2014) is applied to generate question vector according to the hidden layer representation $\boldsymbol{h}$ and the state of the decoder s_{k-1}, which is computed as follows,

$$q_k = \sum_{t=1}^{T} \alpha_{kt} h_t \tag{10}$$

$$\alpha_{kt} = \frac{exp(e_{kt})}{\sum_{j=1}^{T} exp(e_{kj})} \tag{11}$$

$$e_{kt} = v_a^T tanh(W_a s_{k-1} + U_a h_t) \tag{12}$$

$$s_{k-1} = [h_{k-1}^1; h_{k-1}^2] \tag{13}$$

where α_{kt} is the weight of every hidden layer representation h_t of the question sequence, and s_{k-1} is the state of the decoder by concatenating h_{k-1}^1 and h_{k-1}^2 of the decoder. By weighted average of the hidden

layer representation $\boldsymbol{h}$, the decoder can obtain the critical information of the question sequence when generating the next token.

3.4 Act Type Embedding

Dialogue act consists of two parts: the act type that denotes the goal of the response to be generated, such as confirm and recommend, and the slot values which should be satisfied in the response, such as the price range or area of a requested restaurant. The act type information can be very useful to generate the response, however, it has not been fully exploited by SC-LSTM. In order to make the best use of the act type information, we embed the act type to an n-dimentional vector v_d which is randomly initialized and subsequently learned through the iterative training. By concatenating the act type embedding vector v_d and the word vector w_t and feeding them to the network, the act type information is able to influence the generation process at a global level. The computation of input gate, forget gate, output gate, candidate cell and reading gate of the network are updated by the following equations,

$$i_t^n = \sigma(W_{wi}^n[w_t; v_d] + W_{hi}^n h_{t-1}^n + W_{hhi}^n h_t^{n-1} + W_{qi}^n q_t + b_i^n) \quad (14)$$

$$f_t^n = \sigma(W_{wf}^n[w_t; v_d] + W_{hf}^n h_{t-1}^n + W_{hhf}^n h_t^{n-1} + W_{qf}^n q_t + b_f^n) \quad (15)$$

$$o_t^n = \sigma(W_{wo}^n[w_t; v_d] + W_{ho}^n h_{t-1}^n + W_{hho}^n h_t^{n-1} + W_{qo}^n q_t + b_o^n) \quad (16)$$

$$\hat{c}_t^n = tanh(W_{wc}^n[w_t; v_d] + W_{hc}^n h_{t-1}^n + W_{hhc}^n h_t^{n-1} + W_{qc}^n q_t + b_c^n) \quad (17)$$

$$r_t = \sigma(W_{wr}[w_t; v_d] + \sum_n \alpha_n W_{hr}^n h_{t-1}^n). \quad (18)$$

3.5 Training

The objective function is based on the cross entropy error between the predicted token distribution y_t and the gold distribution p_t in the training corpus. And to regularize the slot-value vector d_t, the cost function is modified to the following equation as suggested by (Wen et al., 2015),

$$F(\theta) = \sum_t p_t^\top log(y_t) + \parallel d_T \parallel + \sum_{t=0}^{T-1} \eta \xi^{\|d_{t+1} - d_t\|} \quad (19)$$

where d_T is the slot-value vector at the last time step T , and η and ξ are constants set to 10^4 and 100, respectively. To minimize $\parallel d_T \parallel$ is used to encourage the responses to provide adequate required slots. And the last term $\sum_{t=0}^{T-1} \eta \xi^{\|d_{t+1} - d_t\|}$ discourages the reading gate from turning off more than one bit of slot-value vector in a single time step. The parameters of CA-LSTM network are randomly initialized except for the word embeddings which are initialized by pre-trained 300-dimension word vectors (Pennington et al., 2014). The trade-off weights α are set to 0.5 as mentioned in Section 3.2 and 3.4. The dimension of hidden layer and act type embedding are set to 80 and 20 respectively. The decoder takes the two hidden layer Stacked-LSTM network with a 50% dropout rate. The network is trained with back propagation through time (Werbos, 1990) by treating each user question and system response turn as a mini-batch. AdaDelta (Zeiler, 2012) is applied to optimise the parameters, and the word embeddings are fine tuned through the iterative training. The forward generator and backward reranker are based on the same CA-LSTM network while the parameters are independent.

3.6 Reranking

In the decoding phase, the decoder generates a candidate set of response sequences by randomly sampling the probability distribution of next token. In the reranking phase, the candidate set is reranked by the sentence $score$. As a result, the top-n responses of the candidate set are chosen as the output responses. By combining the cost $F_f(\theta)$ of the forward generator as defined in Eq. (19), the cost $F_b(\theta)$ of the backward reranker (also in Eq. (19)) and the penalty of the error slot, the $score$ is defined as follows,

$$score = -(F_f(\theta) + F_b(\theta) + \lambda \frac{p+q}{N}) \quad (20)$$

where p, q are the number of missing and redundant slots respectively, N is the number of slots required, and λ is set to 100000 to penalize the response with wrong answers.

act types	inform, inform only match inform no match, confirm, hello select, request, reqmore, goodbye
slot names	name, type, *pricerange, postcode price, phone, address, *area, *near *food, *goodformeal, ***kids-allowed**

Table 1: The details about the dialogue act. **bold**=binary slots, *=slots can take the value of *dont care*.

Method	BLEU-4
hdc	0.451
kNN	0.602
classlm	0.627
SC-LSTM	0.731
CA-LSTM	0.775
CA-LSTM+att	0.783
CA-LSTM+att+emb	**0.790**

Table 2: BLEU-4 score of the top 5 responses.

4 Experiments

4.1 Dataset Description

To evaluate the performance of CA-LSTM, we adopt the SF Restaurant dataset as used in (Wen et al., 2015), which is a corpus of a spoken dialogue system providing information about restaurants in San Francisco. It has around 5000 user question and system response turns sampled from about 1000 dialogues. The act types and slot-value pairs are labeled in the dataset. The details about the dialogue act are provided in Table 1. The training, validation and testing set are partitioned in the ratio of 3:1:1. And upsampling w.r.t act type is applied to make the corpus more uniform similar to (Wen et al., 2015).

4.2 Implementation Details

We use Theano (Bergstra et al., 2010) to implement the proposed model. For each dialogue act and input question, we generate 20 responses and select the top 5 responses as the output after reranking. The BLEU-4 metric (Papineni et al., 2002) implemented by NLTK (Bird, 2006) is used for quantitative evaluation. And the references set of the BLEU-4 metric are built by grouping the references of the same dialogue acts after delexicalising the responses and lexicalizing them by the correct values. Since the performance of CA-LSTM depends on initialisation, the results shown below are averaged over 5 randomly initialised CA-LSTM and the corpus are partitioned after random shuffle as well.

4.3 Quantitative Evaluation

We compare our proposed model with several baselines including: the handcrafted generator (hdc), k-nearest neighbour (kNN), class-based LMs (classlm) as proposed by Oh and Rudnicky (2000), the 2-hidder-layer semantically conditioned LSTM network (SC-LSTM) proposed by Wen et al. (2015). For our own method, we experiment with several settings: the basic setting (denoted by CA-LSTM), the Context-Aware LSTM with attention (CA-LSTM+att) which encodes the question vector with an attention mechanism, and the Context-Aware LSTM with attention and act type embeddings (CA-LSTM+att+emb). The result is shown in Table 2. As we can see, the performances of our methods have been greatly improved compared to the baselines shown in the first block (hdc,kNN,classlm and SC-LSTM). By combining more context information (attention and act type embeddings), the performance of CA-LSTM is further improved correspondingly. And the Context-Aware LSTM with attention and act type embeddings (CA-LSTM+att+emb) obtains the best overall performance.

4.4 Human Evaluation

Method	Informativeness	Naturalness
classlm	2.28	2.32
SC-LSTM	2.62	2.63
CA-LSTM	**2.78**	**2.74**

Table 3: Human evaluation for the quality of top 5 responses on two metrics (rating out of 3).

Pref.%	classlm	SC-LSTM	CA-LSTM
classlm	-	16.9	15.7
SC-LSTM	83.1	-	25.3
CA-LSTM	**84.3**	**74.7**	-

Table 4: Pairwise preference among the three systems.

We recruit 10 judges for human evaluation experiments. For each task, the three systems(classlm, SC-LSTM and CA-LSTM with attention and act type embeddings which is denoted by CA-LSTM for simplicity) are used to generate 5 responses. Judges are asked to score each of them in terms of informativeness and naturalness (rating scale is 1,2,3), and also asked to state a preference between any two of them. The informativeness is defined as whether the response provides all the information contained in the DA and the naturalness is defined as whether the response could plausibly have been produced by a human as proposed by Wen et al.(2015). We test 200 DAs and 1000 responses per system in total. The result of human evaluation for the quality of response is shown in Table 3. As can be seen, CA-LSTM outperforms the baseline methods in both metrics of informativeness and naturalness significantly ($p < 0.05$, Student's t-test). Besides, CA-LSTM is preferred by judges as shown in Table 4 where CA-LSTM is much more perferred than SC-LSTM.

4.5 Case Study

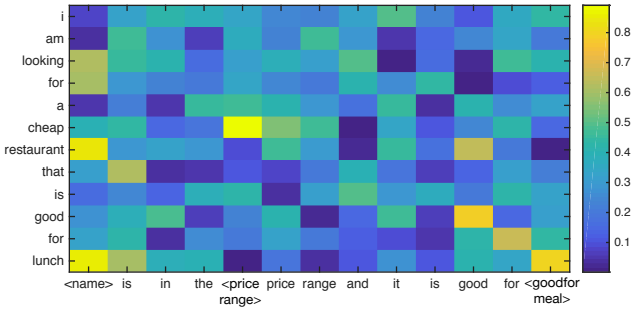

Figure 4: Attention matrix visualization

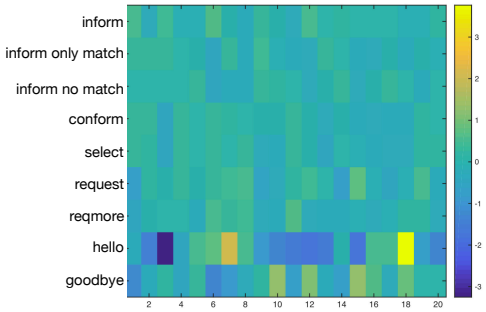

Figure 5: AT embedding visualization

Figure 4 visualizes the attention matrix for a pair of input question and output response. As can be seen, the weights of keywords in the question are strengthened when the decoder generates the relevant tokens, for example the weights of "restaurant", "lunch", "looking", "for" are augmented when the decoder generates the slot token "<name>".

Figure 5 visualizes the learned AT embeddings. The x-axis indicates the dimension index of AT embeddings and the color indicates the value of the corresponding dimension. The similar act types have similar AT embeddings such as "inform" and "inform only match", "request" and "reqmore", while dissimilar act types have different AT embeddings such as "hello" and "goodbye".

CA-LSTM can generate more coherent responses than SC-LSTM for particular act types such as "inform no match". Quantitative evaluation shows that CA-LSTM achieves 0.858 BLEU-4 score compared to 0.772 of SC-LSTM for the act type of "inform no match". For this act type, CA-LSTM can generate negation responses, such as " there is no basque restaurant that allows child -s. ", while SC-LSTM tends to ignore the negation information and generates responses like " there are basque restaurant -s that allow kid -s .".

5 Conclusion and Future Work

In this paper, we have proposed a statistical task-oriented NLG model based on a Context-Aware Long Short-term Memory (CA-LSTM) recurrent network. The network can learn from unaligned data without any heuristics, and it can generate variant responses and provide correct answers in response to the input information. Both quantitative evaluation and human evaluation show that CA-LSTM obtains the state-of-the-art performance. We also reveal the influence of the attention mechanism and act type embeddings with case studies. As future work, we would explore the NLG task in different domains and scenarios which need to consider more context information in the generation process.

6 Acknowledgements

This work was partly supported by the National Basic Research Program (973 Program) under grant No. 2013CB329403, the National Science Foundation of China under grant No.61272227/61332007.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

John Bateman and Renate Henschel. 1999. From full generation to near-templates without losing generality. In *Proceedings of the KI'99 workshop," May I Speak Freely*.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: A cpu and gpu math compiler in python. In *Proc. 9th Python in Science Conf*, pages 1–7.

Steven Bird. 2006. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 69–72. Association for Computational Linguistics.

Stephan Busemann and Helmut Horacek. 2002. A flexible shallow approach to text generation. *Proceedings of the Ninth International Workshop on Natural Language Generation*, pages 238–247.

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. 2005. Bidirectional lstm networks for improved phoneme classification and recognition. In *International Conference on Artificial Neural Networks*, pages 799–804. Springer.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Jiwei Li, Will Monroe, Alan Ritter, and Dan Jurafsky. 2016. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*.

Danilo Mirkovic and Lawrence Cavedon. 2011. Dialogue management using scripts, October 18. US Patent 8,041,570.

Alice H Oh and Alexander I Rudnicky. 2000. Stochastic language generation for spoken dialogue systems. In *Proceedings of the 2000 ANLP/NAACL Workshop on Conversational systems-Volume 3*, pages 27–32. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–43.

Alan Ritter, Colin Cherry, and William B. Dolan. 2011. Data-driven response generation in social media. In *Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John Mcintyre Conference Centre, Edinburgh, Uk, A Meeting of Sigdat, A Special Interest Group of the ACL*, pages 583–593.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2015. Hierarchical neural network generative models for movie dialogues. *arXiv preprint arXiv:1507.04808*.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1577–1586.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Marilyn A Walker, Owen C Rambow, and Monica Rogati. 2002. Training a sentence planner for spoken dialogue using boosting. *Computer Speech & Language*, 16(3):409–433.

Tsung Hsien Wen, Milica Gasic, Nikola Mrksic, Pei Hao Su, David Vandyke, and Steve 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 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1711–1721.

Paul J Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. 2016. Neural generative question answering. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 2972–2978.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.