Incorporating Interlocutor-Aware Context into Response Generation on Multi-Party Chatbots

Cao Liu[1,2], **Kang Liu**[1,2], **Shizhu He**[1,2], **Zaiqing Nie**[3], **Jun Zhao**[1,2]
[1] National Laboratory of Pattern Recognition, Institute of Automation,
Chinese Academy of Sciences, Beijing, 100190, China
[2] University of Chinese Academy of Sciences, Beijing, 100049, China
[3] Alibaba AI Labs, Beijing, 100029, China
{cao.liu, kliu, shizhu.he, jzhao}@nlpr.ia.ac.cn
zaiqing.nzq@alibaba-inc.com

Abstract

Conventional chatbots focus on two-party response generation, which simplifies the real dialogue scene. In this paper, we strive toward a novel task of Response Generation on Multi-Party Chatbot (RGMPC), where the generated responses heavily rely on the interlocutors' roles (e.g., speaker and addressee) and their utterances. Unfortunately, complex interactions among the interlocutors' roles make it challenging to precisely capture conversational contexts and interlocutors' information. Facing this challenge, we present a response generation model which incorporates Interlocutor-aware Contexts into Recurrent Encoder-Decoder frameworks (ICRED) for RGMPC. Specifically, we employ interactive representations to capture dialogue contexts for different interlocutors. Moreover, we leverage an addressee memory to enhance contextual interlocutor information for the target addressee. Finally, we construct a corpus for RGMPC based on an existing open-access dataset. Automatic and manual evaluations demonstrate that the ICRED remarkably outperforms strong baselines.

1 Introduction

Human computer conversation has been an important and challenging task in NLP and AI since the Turing Test was proposed in 1950 (Turing, 1950). Recently, with the rapid growth of social conversation data available on the Internet, data-driven chatbots are able to learn to generate responses directly and have attracted much more attention than before (Li et al., 2016a; Tian et al., 2017).

Researches in this area mostly focus on the dialog with two interlocutors (Maíra Gatti de Bayser et al., 2017). However, the real-life interaction involves a substantial part of Multi-Party Chatbots (MPC, such as internet forum and chat group), which is a form of conversation with multiple interlocutors (Ouchi and Tsuboi, 2016). For example, there are more than three interlocutors $(a_1, a_2, a_3 ... a_m)$ involved in the conversation in Figure 1, and their roles (e.g., speaker and addressee) may change across different dialog turns.

	t	Speaker	Addressee	Utterance
Context	1	Alan (a_1)	Bert (a_2)	the main ubuntu channels require that …
	2	Carl (a_3)	-	i found that worse like a 1 year ago …
	…	…	…	…
	n-1	Carl (a_3)	Jack (a_m)	i have a " dash to dock " extension …
	n	Alan (a_1)	Carl (a_3)	gnome classic is still available …
	n+1	Jack (a_m)	Carl (a_3)	mate is a recent reincarnation i …
		Responding Speaker	Target Addressee	(Generated) Response

Figure 1: An example of Multi-Party Chatbots (MPC). At each turn, a speaker said one utterance to an addressee. There are many interlocutors (e.g., Alan, Bert and so on) in a conversation, where a_i represents interlocutor's ID.

As shown in Figure 1, at each turn, the core issue of MPC is to capture *who* (speaker) talks to *whom* (addressee) about *what* (utterance). In order to obtain responses in MPC, in our best knowledge, previous approaches usually employ a response selection paradigm, which simply selects one response from a set of existing utterances as the final response according to the contexts. Obviously, this paradigm, which could not generate new responses, is not so flexible. In this study, to build a more broadly applicable system, we concentrate on producing new responses word by word, named as Response Generation on Multi-Party Chatbots (RGMPC).

RGMPC is a very challenging task. The primary challenge is that the generated response has strong relevance to the interlocutor's roles, such as the speaker and the addressee. For example, in the same context of Figure 1, what a_1 says to a_2 is different from what a_1 says to a_3 because different addressees (a_2 and a_3) have different information demands. Similarly, as for the same addressee, ut-

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 718–727
Hong Kong, China, November 3-4, 2019.

terances from different speakers may be different because each speaker has personal background knowledge and style of speaking. Moreover, the roles of the same interlocutor may vary across different dialog turns. For instance, in Figure 1, a_3 plays different roles in different dialog turns: speaker in the turn 2 and n-1, addressee in the turn n and n+1.

Therefore, it is very important for RGMPC to capture interlocutor information. Currently, most response generation methods consider only the contextual utterance information (Serban et al., 2016, 2017) but neglect the interlocutor information. Although some researches have exploited the interlocutor information for response generation, they are still suffering from certain critical limitations. Li et al. (2016b) learn a fixed vector for each person from all conversational texts in the training corpus. However, as a global representation, the fixed person vector needs to be trained from large-scale dialogue turns for each interlocutor, and it may have a **sparsity issue** since some interlocutors have very few dialogue turns.

To address the aforementioned problems of RGMPC, this paper incorporates Interlocutor-aware Contexts into a Recurrent Encoder-Decoder model (ICRED) for RGMPC, which is also an end-to-end framework. Specifically, in order to capture interlocutor information, we exploit interactive interlocutor representations learned from current dialog context rather than the fixed person vectors (Li et al., 2016b) obtained from all dialogs in the training corpus. We expect that the learned contextual interlocutor representation could be a good alternative to the fixed person vectors (Li et al., 2016b) due to its ability of alleviating the sparsity issue. Furthermore, from the view of conversation analysis, responses are usually used for answering the addressee's question or expanding the addressee's utterances. Therefore, we originally introduce an addressee memory mechanism to enhance contextual information for the target addressee especially. Finally, both of the interactive interlocutor representation and addressee memory are utilized for decoding response utterances. In particular, the addressee memory is leveraged to capture the addressee information for each generated word dynamically.

In order to prove the effectiveness of the proposed model, we construct a dataset for RGMPC based on an open dataset[1]. Experimental results show that the proposed model is fairly competitive on both automatic and manual evaluations compared with state-of-the-arts.

In brief, the main contributions of the paper are as follows:

(1) We propose an end-to-end response generation model called ICRED which incorporates Interlocutor-aware Contexts into Recurrent Encoder-Decoder framework for RGMPC.

(2) We leverage an addressee memory mechanism to enhance contextual interlocutor information for the addressee.

(3) We construct an open-access dataset for RGMPC. Both automatic and manual evaluations demonstrate that our model is remarkably better than strong baselines in this dataset.

2 Task Formulation

	Data	Notation
Input	Context	$\mathcal{C} = [(a^t_{spk}, a^t_{adr}, \mathbf{u}^t)]^n_{t=1}$
	Responding Speaker	a^{n+1}_{spk} (or a_{res})
	Target Addressee	a^{n+1}_{adr} (or a_{tgt})
Output	Response	$\mathbf{u}^{n+1}$ (or $\{r_j\}^{L_r}_{j=1}$)

Table 1: Notations for RGMPC.

On multi-party chatbots, lots of interlocutors talk about one or more topics. At each dialogue turn (or time step) t, there is a speaker (a^t_{spk}), who may talk something ($\mathbf{u}^t$) to a specific addressee (a^t_{adr}), while the others are observers. As shown in Table 1, given the context $\mathcal{C}$ of previous n dialog turns, the responding speaker a_{res} and the target addressee a_{tgt} at time step n+1, the task of RGMPC aims to automatically generate the next utterance $\mathbf{u}^{n+1}$ as the final response. Here, $\mathcal{C}$ is a list ordered by the time step t: $\mathcal{C} = [\mathcal{C}^t]^n_{t=1} = [(a^t_{spk}, a^t_{adr}, \mathbf{u}^t)]^n_{t=1}$, where $\mathcal{C}^t$ means a^t_{spk} says $\mathbf{u}^t$ to a^t_{adr} at time step t, n is the maximum number of previous dialog turns in a context. $\mathbf{u}^t = (w^t_1, w^t_2 ... w^t_{L_u})$ is the input utterance (word sequence) at time step t, where L_u is the number of maximum words in utterances.

3 Methodology

The overview of the proposed ICRED for RGMPC is shown in Figure 2 along with its caption. The details are as follows.

[1]The dataset is available at https://www.dropbox.com/s/4chh64yaxajh0j7/RGMPC.zip?dl=0

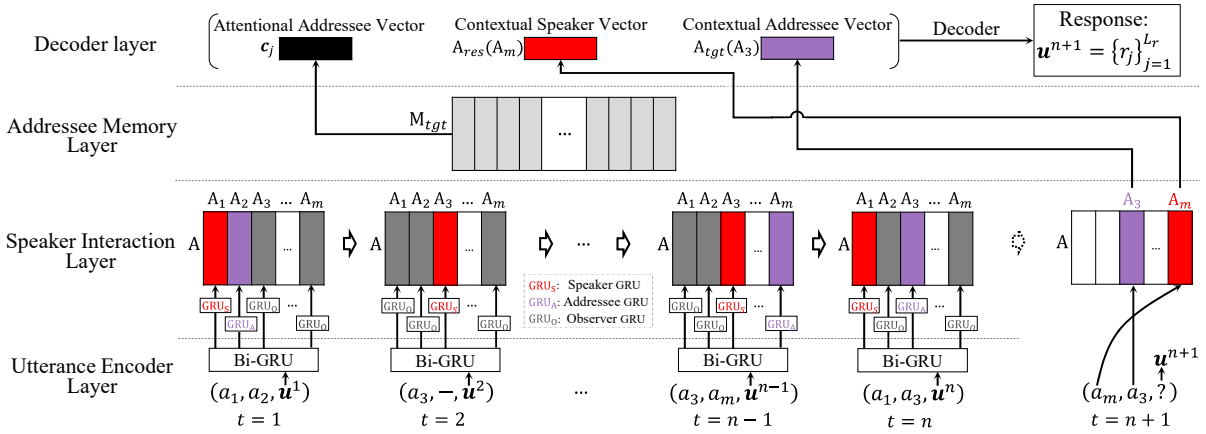

Figure 2: Overall structure of the proposed ICRED for RGMPC. At each time step t, $(a_i, a_j, \mathbf{u}^t)$ means that a speaker a_i said an utterance $\mathbf{u}^t$ to an addressee a_j, where the time step t is denoted on the bottom, and the superscript t may be omitted for brevity. Our ICRED includes: ① Utterance Encoder Layer: encoding each utterance ($\mathbf{u}^t$) into distributed vectors; ② Speaker Interaction Layer: capturing interactive interlocutor information from contexts, and it updates all interlocutors' representation by different GRUs according to their roles at each time step, where the embedding for an interlocutor a_i is obtained by extracting the i-th column (A_i) from the interlocutor embedding matrix A; ③ Addressee Memory Layer: enhancing contextual information for the target addressee (a_3); ④ Decoder Layer: generating responses.

3.1 Utterance Encoder Layer

The utterance encoder layer transforms input utterance into distributional representations. We leverage the bi-directional Gated Recurrent Units (GRU) (Cho et al., 2014) to capture the long-term dependency. For an utterance $\mathbf{u}^t = (w_1^t, w_2^t ... w_{L_u}^t)$ at time step t, the concatenated representation for hidden states in bi-directions is denoted as $\mathbf{h}_i^t = [\overrightarrow{\mathbf{h}}_i^t, \overleftarrow{\mathbf{h}}_{L_u-i+1}^t]$, where $\mathbf{h}_i^t$ is considered as the contextual word representation of the input word w_i^t. The state ($\mathbf{h}_{L_u}^t$) of the last word is treated as the representation of the utterance at time step t, which is denoted as $\mathbf{h}^t$, and it could be sent to the speaker interaction layer for updating contextual representation.

3.2 Speaker Interaction Layer

The speaker interaction layer is leveraged to obtain the interlocutor information in the context. Similar to the Speaker Interaction RNNs (Zhang et al., 2018), we utilize the interactive speaker encoder for RGMPC.

As shown in Figure 2, an interlocutor embedding matrix A is used to record all interlocutors' representation, and A is initiated with a zero matrix. Each column of A corresponds to an interlocutor's embedding: $A_i = A[*, a_i]$, where A_i is the embedding for the interlocutor a_i. The speaker interaction layer updates the entire interlocutors' embeddings at each time step based on their roles (speaker, addressee or observer). Embeddings for the speaker, addressee and observer are updated by following role-differentiated GRUs: GRU_S, GRU_A and GRU_O, respectively.

$$A_{spk}^t = \text{GRU}_S(A_{spk}^{t-1}, \mathbf{h}^t) \quad (1)$$

$$A_{adr}^t = \text{GRU}_A(A_{adr}^{t-1}, \mathbf{h}^t) \quad (2)$$

$$A_{obv}^t = \text{GRU}_O(A_{obv}^{t-1}, \mathbf{h}^t) \quad (3)$$

where A_{spk}^t (A_{adr}^t / A_{obv}^t) is the embedding for the speaker (addressee / observer) at time step t, and $\mathbf{h}^t$ is the utterance representation obtained from the utterance encoder layer. Take the first time step "$(a_1, a_2, \mathbf{u}^1)$" in Figure 2 as an example, when a_1 says $\mathbf{u}^1$ to a_2, the speaker's (a_1's) embedding A_1 is updated by the speaker GRU—GRU_S, and the addressee's (a_2's) embedding A_2 is updated by the addressee GRU—GRU_A, while other interlocutors' embeddings are updated by the observer GRU—GRU_O. Note that the addressee may be missing (such as "$(a_3, -, \mathbf{u}^2)$" at time step 2 in Figure 2), where embeddings for all interlocutors except for the speaker are updated by the observer GRU. The interlocutor embedding matrix (A) is updated up to the maximum time step n. The final interlocutor embedding matrix is used in decoding.

3.3 Addressee Memory Layer

The interlocutor embedding matrix is updated by utterance representations and interlocutor's roles,

so it captures interlocutor's context on the utterance level. In fact, contextual word representation is important for response generation, too. A context contains consecutive utterances, and each utterance is a word sequence. Therefore, memorizing all contextual word representations in the entire context is complex, and it is difficult to work on large-scale utterances in one context.

Intuitively, from the view of conversational analysis, responses are usually used for answering the addressee's question or expanding the addressee's utterances. Therefore, we design an addressee memory layer, which only memorizes the contextual word representations (noted as M_{tgt}) in the last utterance said by the target addressee, and the contextual representation for each word is obtained from the utterance encoder layer. Take "$(a_m, a_3, ?)$" at time step n+1 in Figure 2 as an example, $\mathbf{u}^{n-1}$ is the last utterance said by the target addressee a_3 because of "$(a_3, a_m, \mathbf{u}^{n-1})$" at time step n-1, so the addressee memory layer merely memorizes contextual word representation $M_{tgt} = [\mathbf{h}_1^{n-1}, \mathbf{h}_2^{n-1}, ..., \mathbf{h}_{L_u}^{n-1}]$ from the utterance $\mathbf{u}^{n-1}$, where $\mathbf{h}_i^{n-1}$ is obtained from Section 3.1.

3.4 Decoder Layer

The decoder is responsible for generating target sequences. Different from a single contextual representation in previous work (Serban et al., 2017), the speaker interaction layer is able to capture different interlocutor information from contexts (e.g., personal background knowledge and style of speaking for the responding speaker, special information demands for the target addressee). Moreover, the addressee memory layer records contextual word representation for the target addressee. Therefore, we extract **contextual speaker vector** A_{res} for the responding speaker a_{res} from the final interlocutor embedding matrix A (e.g., the responding speaker's embedding obtained by $A_m = A[*, a_m]$ for the responding speaker a_m in Figure 2). Similarly, **contextual addressee vector** A_{tgt} for the target addressee is also extracted from A. However, A_{res} and A_{tgt} keep same for each generated word. In order to capture dynamic information for different generated words, we leverage an attention mechanism to selectively reads different contextual word representations from the addressee memory. For each target word, the decoder attentively reads the contextual word representation as follows:

$$\mathbf{c}_j = \sum_{k=1}^{L_u} \alpha_{jk} M_{tgt}[*, k]; \tag{4}$$

$$\alpha_{jk} = \frac{e^{\rho(\mathbf{s}_{j-1}, M_{tgt}[*,k])}}{\sum_{k'} e^{\rho(\mathbf{s}_{j-1}, M_{tgt}[*,k'])}} \tag{5}$$

where $\mathbf{c}_j$ is the **attentional addressee vector**, $M_{tgt}[*, k]$ is the contextual word representation for the k-th word in the addressee memory, and $\mathbf{s}_j$ represents the hidden state in decoding GRU. A function ρ is leveraged to compute the attentive strength, which is calculated by a projected matrix to connect $\mathbf{s}_{j-1}^T$ and $M_{tgt}[*, k]$. Finally, the attentional addressee vector $\mathbf{c}_j$, contextual speaker vector A_{res} and contextual addressee vector A_{tgt} are concatenated to estimate the probability for predicted words:

$$p(r_j | r_{<j}, a_{spk}, a_{tgt}, \mathcal{C}) = p(r_j | r_{j-1}, \mathbf{c}_j, A_{res}, A_{tgt}, \mathbf{s}_j) \tag{6}$$

$$\mathbf{s}_j = \mathrm{GRU}_{dec}(\mathbf{s}_{j-1}, [\mathbf{c}_j, A_{res}, A_{tgt}, \mathbf{x}_{j-1}]) \tag{7}$$

where $\mathbf{s}_j$ is the hidden state of the decoding GRU—$\mathrm{GRU}_{\mathrm{dec}}$. $\mathbf{x}_j$ is the word vector of the predicted target word r_j, and r_j is typically performed by a $softmax$ classifier over a settled vocabulary based on word embedding similarity.

3.5 Learning

The proposed ICRED for RGMPC is totally differentiable, and it can be optimized in an end-to-end manner using back-propagation. Given the context $\mathcal{C}$, responding speaker a_{res}, target addressee a_{tgt} and target word sequence $\{r_j\}_{j=1}^{L_r}$, the objective function is to minimize the loss function:

$$\mathcal{L} = \frac{-1}{L_r} \sum_{j=1}^{L_r} log[p(r_j | r_{<j}, \mathcal{C}, a_{res}, a_{tgt}] + \lambda \mathcal{L}_2 \tag{8}$$

It contains a negative log-likelihood for generated responses and L2 regularization ($\mathcal{L}_2$), where λ is a hyperparameter for $\mathcal{L}_2$.

4 Experiment

4.1 Dataset

Our dataset is constructed based on the Ubuntu multi-party chatbot corpus[2], which has been widely used as the evaluation dataset for the response selection task (Ouchi and Tsuboi, 2016; Zhang

[2]https://github.com/hiroki13/response-ranking

	Total	Train	Dev	Test
# Contexts	423.5K	338.9K	42.3K	42.3K
# Speaker	35.3K	33.5K	15.6K	15.6K
# Addressee	23.4K	22.4K	10.8K	10.8K
# Vocab	276.1K	254.8K	82.2K	82.0K
# Tokens	26.3M	21.0M	2.62M	2.62M
Avg. Tok/Ctx	51.4	51.5	51.4	51.3
Avg. Tok/Res	10.6	10.6	10.7	10.6

Table 2: Data statistics. "#" means number, and "Avg. Tok/Ctx (or Res)" is the number of tokens per context (or response).

et al., 2018). The original data comes from the Ubuntu IRC chat log, where each line consists of (Time, Speaker, Utterance). If the addressee is explicitly mentioned in the utterance, it is extracted as the addressee. Otherwise, all interlocutors except the speaker are observers. Considering that generating new responses in this paper is more complicated than retrieving responses, the generative task requires higher-quality data. We suppose that the responding speaker and target addressee have appeared in the context, where the contextual window is set to 5. Moreover, the words are tokenized by NLTK, and some general responses are removed by human rules[3]. Finally, we randomly split the dataset into Train/Dev/Test (8:1:1), and it is publicly available[1]. The detailed statistics of the dataset are shown in Table 2.

4.2 Implement Details

In order to keep our model comparable to other typical existing methods, we keep the same parameters and experimental environments for ICRED and the comparative models. We take a maximum of 20 words for the utterance. The word vector dimension is 300 and it is initialized with the public released fasttext[4] pre-trained on Wikipedia. The utterance and interlocutor are encoded by 512-dimensional and 1024-dimensional vectors, respectively. The joint loss function with 0.0001 L2 weight is minimized by an Adam optimizer. We implemented all the models with Tensorflow on an NVIDIA TITAN X GPU.

4.3 Automatic Evaluation Metrics

Automatic evaluations (AEs) for Natural Language Generation (NLG) is a challenging and under-researched problem (Novikova et al., 2017). Following (Liu et al., 2018), we leverage two referenced measurements (BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004)[5]) for automatic evaluations. Considering that current data-driven approaches tend to generate short and generic (meaningless) responses, two unreferenced ("intrinsic") metrics are also leveraged to the evaluation. The first one is the average length of responses, which is an objective and surfaced metric reflected the substance of responses (Mou et al., 2016; He et al., 2017a). The other one is the number of nouns[6] per response (Liu et al., 2018), which shows the richness of responses since nouns are usually content words. Note that the unreferenced metrics could enrich the evaluations, though they are weak metrics. The detailed results and analyses are shown as follows.

4.4 The Effectiveness of ICRED for RGMPC

Model	Referenced		Unreferenced	
	BLEU	ROUGE	Length	#Noun
Seq2Seq	8.86	7.62	9.48	1.24
Persona Model	9.12	7.38	11.04	1.29
VHRED	9.38	7.65	10.25	1.55
ICRED (ours)	**10.63**	**8.73**	**11.34**	**1.68**

Table 3: Overall comparisons of ICRED.

Comparison Methods. We compared ICRED with the following methods:

(1) Seq2Seq (Sutskever et al., 2014): Seq2Seq is one of the mainstream methods for text generation. In order to capture as much information as possible, the input sequence is all utterances concatenated in order in a context.

(2) Persona Model (Li et al., 2016b): The persona-based model modified a Seq2Seq to encode a global vector for each interlocutor that appears in the training data, and it could alleviate the issue of speaker consistency for response generation.

(3) VHRED (Serban et al., 2017): VHRED is essentially a conditional variational auto-encoder with hierarchical encoders, and it extends HRED (Serban et al., 2016) by adding a high-dimensional latent variable for utterances.

Comparative Results. Table 3 demonstrates overall comparisons of ICRED. We can clearly obtain the following observations:

[3]We list some general responses, such as containing "*i don't know*", "*you are welcome*".

[4]https://github.com/facebookresearch/fastText

[5]Implemented by https://github.com/Maluuba-/nlg-eval. BLEU and ROUGE are transformed into percentages (%).

[6]NLTK is utilized for part-of-speech tagging.

Interlocutor's Dialogue Turns	Persona Model		ICRED (ours)	
	BLEU	ROUGE	BLEU	ROUGE
[0, 100]	8.47	6.72	10.63	8.60
(100, 1000]	8.87	7.14	10.50	8.61
(1000, 5000]	9.48	7.74	**10.77**	**8.90**
(5000, +∞)	**9.51**	**7.80**	10.60	8.79

Table 4: Performances on sparse and plentiful learning data with different numbers of interlocutor's dialogue turns, where the test data is divided into different intervals according to the number of dialogue turns in training dataset said by target addressee (named as interlocutor's dialogue turns).

(1) ICRED obtains the highest performance on all metrics (marked as **bold**), and it indicates that incorporating interlocutor-aware context into RGMPC contributes to generating better responses.

(2) Although the persona-based model utilizes interlocutor information, it performs poorly. The average dialogue turn for the interlocutor is more than 5000 in (Li et al., 2016a), while there is less than 100 dialogue turns per interlocutor in our dataset. Therefore, it is hard to learn a global vector for each interlocutor from the sparse corpus. In contrast, our ICRED performs well on such a sparse corpus (details in Section 4.5).

(3) VHRED brings slight improvements over the Seq2Seq and persona-base model. Even that VHRED enhances the contextual information by a high-dimensional latent variable, VHRED is still remarkably worse than ICRED because VHRED neglects the interlocutor information.

4.5 The Effect of Sparse Data on ICRED

Comparison Settings. Persona model (Li et al., 2016b) may have a sparsity issue since some interlocutors have very few dialogue turns. To investigate whether ICRED has the sparsity issue or not, we divide the test data into four intervals according to the number of training dialogue turns said by the target addressee (called interlocutor dialogue turns), where small turns represent sparse learning data (e.g., "[0, 100]") and large turns mean plentiful learning data (e.g., "(5000, +∞)").

Comparative Results. Table 4 reports the performances of persona model and ICRED on different interlocutor's dialogue turns for learning. We can clearly see that the persona model has a sparsity issue: it performs very poorly on sparse learning data (e.g., BLEU score = 8.47 on "[0, 100]") while it achieves good performances on plentiful learning data (e.g., BLEU score = 9.51 on "(5000, +∞)"), which demonstrates that the fixed person vectors in the persona model need to be learned from large-scale training data for each interlocutor. In contrast, ICRED exploits interactive interlocutor representation learned from current dialog context rather than the fixed person vectors obtained from all training dialog utterances. Therefore, ICRED has no sparsity issues and it performs closely on sparse and plentiful learning data.

4.6 Ablation Study for Model Components

Model	Referenced		Unreferenced	
	BLEU	ROUGE	Length	#Noun
ICRED	**10.63**	**8.73**	**11.34**	**1.68**
w/o Adr_Mem	10.25	8.23	10.73	1.27
w/o Ctx_Spk_Vec	10.13	8.22	10.86	1.59
w/o Ctx_Adr_Vec	9.95	8.18	10.93	1.26

Table 5: Ablation Experiments by removing the main components.

Comparison Settings. In order to validate the effectiveness of model components, we have tried to remove some main components in decoding as follows. (1) w/o Adr_Mem: without the addressee memory, such as removing $\mathbf{c}_j$ in Equation 6-7; (2) w/o Ctx_Spk_Vec: without the contextual speaker vector, such as removing A_{res} in Equation 6-7; (3) w/o Ctx_Adr_Vec: without the contextual addressee vector, such as removing A_{tgt} in Equation 6-7.

Comparative Results. Results of the ablation study are shown in Table 5. We can see that removing any component causes obvious performance degradation. In particular, "w/o Ctx_Adr_Vec" performs the worst on almost all of the metrics, which demonstrates the importance of contextual information for the target addressee.

4.7 The Effectiveness of Addressee Memory

Memory Type	Referenced		Unreferenced	
	BLEU	ROUGE	Length	#Noun
addressee memory	**10.63**	8.73	11.34	**1.68**
all utterance memory	10.39	**8.78**	**11.38**	1.37
latest memory	10.43	8.40	10.16	1.28
speaker memory	10.03	8.28	10.72	1.66
w/o memory	10.25	8.23	10.73	1.27

Table 6: Performances over different memory types.

Comparison Settings. In order to demonstrate the effectiveness of the addressee memory, we

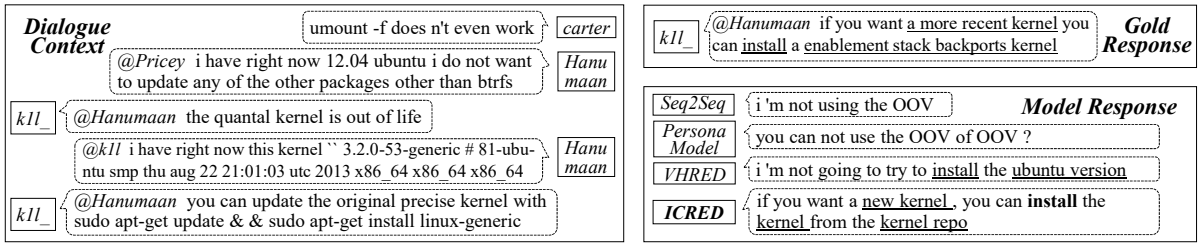

Figure 3: An example of different model responses for the same dialogue context. The input dialogue context is on the left. The gold (referenced) response and model responses are on the top right and bottom right, respectively. The rounded rectangle is the message box, where the *italic* behind "@" is the addressee, and the solid-line box near to the message box represents the speaker or model.

change the memory type, and then the attention model in Equation 5 is based on the new memory. The comparison settings are shown as follows. (1) addressee memory: memorizing contextual word representations in the last utterance said by the *target addressee* (e.g., $\mathbf{u}^{n-1}$ in Figure 2); (2) all utterance memory: memorizing contextual word representations in *all* utterances of the context (e.g., $\mathbf{u}^1$ to $\mathbf{u}^n$ in Figure 2); (3) latest memory: memorizing contextual word representations of the *latest* utterance in the context (e.g., the latest utterance $\mathbf{u}^n$ in Figure 2); (4) speaker memory: memorizing contextual word representations in the last utterance said by the *responding speaker*; (5) w/o memory: without any memory.

Comparative Results. We report the results of different memory types as shown in Table 6. It can see that our method, the addressee memory, achieves the best or near-best performances on all metrics. Although memorizing all utterances is competitive, the complexity of all utterance memory is n times compared with the one in the addressee memory, where n is the number of utterances in a context. The speaker memory performs closely to without memory, which indicates that not all memories can improve the performance.

4.8 Manual Evaluations

Besides automatic evaluations, we employ manual evaluations (MEs), which is important for response generation. Similar to (He et al., 2017b; Zhou et al., 2018), and we select three metrics for MEs, which measure the following aspects. (1) Fluency: measuring whether responses are grammatically correct or wrong. (2) Consistency: measuring whether responses are coherent to the context or not. (3) Informativeness: measuring how much informational (knowledgeable) content obtained from the responses.

Model	Flu.	Con.	Inf.
ICRED vs. Seq2Seq	77.25	83.69	84.35
ICRED vs. Persona.	78.44	80.41	82.35
ICRED vs. VHRED	73.20	81.29	79.47

Table 7: Manual evaluations (%) with fluency (Flu.), consistency (Con.) and informativeness (Inf.). The Score is the percentage that ICRED wins baselines after removing the "tie" pairs.

We conduct a pair-wise comparison between the response generated by ICRED and the one for the same input by three typical baselines. We sample 100 responses from each compared methods. Two curators judge (win, tie and lose) between these two methods. The Cohen Kappa of inter-annotator statistics is 0.750, 0.658 and 0.580 for the fluency, consistency and informativeness, respectively. As shown in Table 7, the score is the percentage that ICRED wins baselines after removing the "tie" pairs, and we can obtain that ICRED is significantly (sign test, p-value < 0.005) superior to all baselines on any metric. It demonstrates our model is able to deliver more fluent, consistent and informative responses.

4.9 Case Study

Figure 3 shows an example of responses on different models for the same dialogue context. It is clearly observed that our model (**ICRED**) generates more fluent, consistent and knowledgeable (marked as underline) responses compared to baselines. In particular, the response given by ICRED "*if you want a new kernel , you can install the kernel from the kernel repo*", not only explains the *reason for kernel installation* but also suggests a *source of the installation*. It fully captures the context and then produces a fluent, consistent and knowledgeable response, which is semantically similar to the gold one.

4.10 Discussion

Interlocutor Prediction and RGMPC. The above methods assume that the responding speaker and target addressee are given for RGMPC. Though the speaker and the addressee could be obtained in some situations (e.g., extracted from chat logs), it is still a researchable task to interlocutor prediction. There have been some researches to predict either the responding speaker or the target addressee based on the given textual contexts or multimodal information (Akhtiamov et al., 2017a; Meng et al., 2017; Akhtiamov et al., 2017b). Nevertheless, in order to obtain the interaction between interlocutor prediction and RGMPC, we further design a joint model for RGMPC and interlocutor prediction. Note that both the speaker and the addressee are predicted based on textual contexts, simultaneously. Firstly, the responding speaker is predicted from contexts:

$$p(a_{res}|\mathcal{C}) = \sigma([\mathbf{h}_{\mathcal{C}}; \mathbf{h}^n_{L_u}] \cdot \mathbf{W} \cdot A_{res}) \quad (9)$$

where $\mathbf{h}_{\mathcal{C}}$ is a summary contextual vector, which is max-pooled by the final interlocutor embedding matrix (A), and $\mathbf{h}^n_{L_u}$ is the hidden state of the last utterance. $\mathbf{W}$ is a projected matrix. a_{res} and A_{res} are the ID and the embedding of the responding speaker, respectively. The responding speaker is predicted by a *softmax* classifier based on the embedding similarity, and the target addressee is obtained in the same way. Secondly, the predicted interlocutors replace the gold ones for the addressee memory and extracting interlocutor's embeddings from A. Finally, the interlocutor prediction loss is added to the response generation loss for training. Table 8 shows the response generation performance on the situation that responding interlocutors are given and predicted. We can observe that:

(1) The overall performance on predicted interlocutors ("* / *" in Table 8) is slightly worse than the one with gold interlocutors (the first line in Table 8). Nevertheless, "* / *" still outperforms the strongest baseline (VHRED in Table 3).

(2) The correctness of interlocutor prediction has a significant impact on response generation performance. It performs the best when the responding speaker and the target addressee are predicted correctly. "False / False" (both are mispredicted) obtains the worst performance on the referenced metrics. These results demonstrate that both responding speaker and target addressee contribute to generating better responses.

Person	Speaker / Addressee	Referenced		Unreferenced	
		BLEU	ROUGE	Length	#Noun
Gold	True / True	10.63	8.73	11.34	1.68
Predict	$*$ / $*$	9.62	7.88	11.99	1.44
	True / True	**10.05**	**8.36**	12.04	1.43
	True / $*$	9.91	8.18	11.95	1.43
	$*$ / True	9.89	8.21	11.97	1.43
	False / False	9.20	7.41	**12.18**	**1.47**

Table 8: Performance on learning interlocutor prediction and RGMPC. "True" and "False" means right and wrong interlocutor, respectively. "*" represents both "True" and "False". The correctness of the responding speaker and target addressee is segmented by "/". For example, "True / *" means that the responding speaker is right, and the target addressee is right or wrong.

(3) Surprisingly, the unreferenced metrics perform well on "False / False". One possible reason is that the wrong interlocutors also capture rich contexts, and it generates long and meaningful responses but with a weak correlation to the gold interlocutors. Therefore, it achieves very poor performance on the referenced metrics.

5 Related Work

Our work is inspired by a large number of applications utilizing recurrent encoder-decoder frameworks (Cho et al., 2014) on NLP tasks such as machine translation (Bahdanau et al., 2015) and text summarization (Chopra et al., 2016). Recently, many researches extend the encoder-decoder framework on response generation. HRED (Serban et al., 2016) utilizes hierarchical encoder to capture the context. VHRED (Serban et al., 2017) extends HRED by adding a high-dimensional latent variable for utterances. These researches demonstrate the importance of contexts on response generation.

Our work is also inspired by researches on multi-party chatbots. Dielmann and Renals (2008) automatically recognize dialogue acts in multi-party speech conversations. Recently, some studies focus on the three elements (speaker, addressee, response) on multi-party chatbots. Meng et al. (2017) introduce speaker classification as a surrogate task. Addressee selection is researched by (Akhtiamov et al., 2017b). Some researches strive to the response selection (Ouchi and Tsuboi, 2016; Zhang et al., 2018). However, the response selection heavily relies on the candidates, and it can not generate new responses in new dialogue contexts. Response generation could solve

this problem. Li et al. (2016b) learn fixed person vector for response generation. Unfortunately, it needs to be obtained from large-scale dialogue turns, which has a sparsity issue: some interlocutors have very little dialog data. Differently, our model has no such restrictions.

6 Conclusion

In this study, we formalize a novel task of Response Generation for Multi-Party Chatbots (RGMPC) and propose an end-to-end model which incorporates Interlocutor-aware Contexts into Recurrent Encoder-Decoder frameworks (ICRED) for RGMPC. Specifically, we employ interactive speaker models to capture contextual interlocutor information. Moreover, we leverage an addressee memory mechanism to enrich contextual information. Furthermore, we propose to predict both the speaker and the addressee when generating responses. Finally, we construct a corpus for RGMPC. Experimental results demonstrate the ICRED remarkably outperforms strong baselines on automatic and manual evaluation metrics.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No.61533018), the Natural Key R&D Program of China (No.2017YFB1002101), the National Natural Science Foundation of China (No.61702512) and the independent research project of National Laboratory of Pattern Recognition. This work was also supported by CCF-DiDi BigData Joint Lab.

References

Oleg Akhtiamov, Maxim Sidorov, Alexey A. Karpov, and Wolfgang Minker. 2017a. Speech and text analysis for multimodal addressee detection in human-human-computer interaction. In *Proceedings of INTERSPEECH*, pages 2521–2525.

Oleg Akhtiamov, Dmitrii Ubskii, Evgeniia Feldina, Alexey Pugachev, Alexey Karpov, and Wolfgang Minker. 2017b. Are you addressing me? multimodal addressee detection in human-human-computer conversations.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *Proceedings of ICLR*.

Paulo Rodrigo Cavalin Maíra Gatti de Bayser, Renan Souza, Alan Braz, Heloisa Candello, Claudio S. Pinhanez, and Jean-Pierre Briot. 2017. A hybrid architecture for multi-party conversational systems. *CoRR*, abs/1705.01214.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of EMNLP*, pages 1724–1734.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of NAACL*, pages 93–98.

Alfred Dielmann and Steve Renals. 2008. Recognition of dialogue acts in multiparty meetings using a switching dbn. *IEEE transactions on audio, speech, and language processing*, pages 1303–1314.

He He, Anusha Balakrishnan, Mihail Eric, and Percy Liang. 2017a. Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings. In *Proceedings of ACL*, pages 1766–1776.

Shizhu He, Cao Liu, Kang Liu, and Jun Zhao. 2017b. Generating natural answers by incorporating copying and retrieving mechanisms in sequence-to-sequence learning. In *Proceedings of ACL*, pages 199–208.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proceedings of NAACL*, pages 110–119.

Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. 2016b. A persona-based neural conversation model. In *Proceedings of ACL*, pages 994–1003.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proceedings of ACL workshop*, page 10.

Cao Liu, Shizhu He, Kang Liu, and Jun Zhao. 2018. Curriculum learning for natural answer generation. In *Proceedings of IJCAI*, pages 4223–4229.

Zhao Meng, Lili Mou, and Zhi Jin. 2017. Hierarchical rnn with static sentence-level attention for text-based speaker change detection. In *Proceedings of CIKM*, pages 2203–2206.

Lili Mou, Yiping Song, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2016. Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation. In *Proceedings of COLING*, pages 3349–3358.

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017. Why we need new evaluation metrics for nlg. In *Proceedings of EMNLP*, pages 2241–2252.

Hiroki Ouchi and Yuta Tsuboi. 2016. Addressee and response selection for multi-party conversation. In *Proceedings of EMNLP*, pages 2133–2143.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318.

Iulian V. Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of AAAI*, pages 3776–3783.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of AAAI*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of NIPS*, pages 3104–3112.

Zhiliang Tian, Rui Yan, Lili Mou, Yiping Song, Yansong Feng, and Dongyan Zhao. 2017. How to make context more useful? an empirical study on context-aware neural conversational models. In *Proceedings of ACL*, pages 231–236.

Turing. 1950. Computing machinery and intelligence. *Mind*, pages 433–460.

Rui Zhang, Honglak Lee, Lazaros Polymenakos, and Dragomir Radev. 2018. Addressee and response selection in multi-party conversations with speaker interaction rnns. In *Proceedings of AAAI*.

Hao Zhou, Tom Young, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. 2018. Commonsense knowledge aware conversation generation with graph attention. In *Proceedings of IJCAI*, pages 4623–4629.