Retrospective and Prospective Mixture-of-Generators for Task-oriented Dialogue Response Generation

Pei, J.; Ren, P.; Monz, C.; de Rijke, M.

DOI
10.3233/FAIA200339

Publication date
2020

Document Version
Author accepted manuscript

Published in
ECAI 2020

Citation for published version (APA):

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.

UvA-DARE is a service provided by the library of the University of Amsterdam (https://dare.uva.nl)
Retrospective and Prospective Mixture-of-Generators for Task-oriented Dialogue Response Generation

Jiahuan Pei and Pengjie Ren and Christof Monz and Maarten de Rijke

Abstract. Dialogue response generation (DRG) is a critical component of task-oriented dialogue systems (TDSs). Its purpose is to generate proper natural language responses given some context, e.g., historical utterances, system states, etc. State-of-the-art work focuses on how to better tackle DRG in an end-to-end way. Typically, such studies assume that each token is drawn from a single distribution over the output vocabulary, which may not always be optimal. Responses vary greatly with different intents, e.g., domains, system actions. We propose a novel mixture-of-generators network (MoGNet) for DRG, where we assume that each token of a response is drawn from a mixture of distributions. MoGNet consists of a chair generator and several expert generators. Each expert is specialized for DRG w.r.t. a particular intent. The chair coordinates multiple experts and combines the output they have generated to produce more appropriate responses. We propose two strategies to help the chair make better decisions, namely, a retrospective mixture-of-generators (RMoG) and a prospective mixture-of-generators (PMoG). The former only considers the historical expert-generated responses until the current time step while the latter also considers possible expert-generated responses in the future by encouraging exploration. In order to differentiate experts, we also devise a global-and-local (GL) learning scheme that forces each expert to be specialized towards a particular intent using a local loss and trains the chair and all experts to coordinate using a global loss. We carry out extensive experiments on the MultiWOZ benchmark dataset. MoGNet significantly outperforms state-of-the-art methods in terms of both automatic and human evaluations, demonstrating its effectiveness for DRG.

1 INTRODUCTION

Task-oriented dialogue systems (TDSs) have sparked considerable interest due to their broad applicability, e.g., for booking flight tickets or scheduling meetings [32, 34]. Existing TDS methods can be divided into two broad categories: pipeline multiple-module models [2, 5, 34] and end-to-end single-module models [11, 30]. The former decomposes the TDS task into sequentially dependent modules that are addressed by separate models while the latter proposes to use an end-to-end model to solve the entire task. In both categories, there are many factors to consider in order to achieve good performance, such as user intent understanding [31], dialogue state tracking [37], and dialogue response generation (DRG). Given a dialogue context (dialogue history, states, retrieved results from a knowledge base, etc.), the purpose of DRG is to generate a proper natural language response that leads to task-completion, i.e., successfully achieving specific goals, and that is fluent, i.e., generating natural and fluent utterances.

Recently proposed DRG methods have achieved promising results (see, e.g., LaRLAttnGRU [36]). However, when generating a response, all current models assume that each token is drawn from a single distribution over the output vocabulary. This may be unreasonable because responses vary greatly with different intents, where intent may refer to domain, system action, or other criteria for partitioning responses, e.g., the source of dialogue context [24]. To support this claim, consider the training set of the Multi-domain Wizard-of-Oz (MultiWOZ) benchmark dataset [4], where 67.4% of the dialogues span across multiple domains and all of the dialogues span across multiple types of system actions. We plot the density of the relative token frequency distributions in responses of different intents over the output vocabulary in Fig. 1. Although there is some overlap among distributions, there are also clear differences. For example, when generating the token [entrance], it has a high probability of being drawn from the distributions for the intent of booking an attraction, but not from booking a taxi. Thus, we hypothesize that a response should be drawn from a mixture of distributions for multiple intents rather than from a single distribution for a general intent.

We propose a mixture-of-generators network (MoGNet) for DRG,
which consists of a chair generator and several expert generators. Each expert is specialized for a particular intent, e.g., one domain, or one type of action of a system, etc. The chair coordinates multiple experts and generates the final response by taking the utterances generated by the experts into consideration. Compared with previous methods, the advantages of MoGNet are at least two-fold: First, the specialization of different experts and the use of a chair for combining the outputs breaks the bottleneck of a single model [10, 19]. Second, it is more easily traceable: we can analyze who is responsible when the model makes a mistake and generates an inappropriate response.

We propose two strategies to help the chair make good decisions, i.e., retrospective mixture-of-generators (RMoG) and prospective mixture-of-generators (PMoG). RMoG only considers the retrospective utterances generated by the experts, i.e., the utterances generated by all the experts prior to the current time step. However, a chair without a long-range vision is likely to make sub-optimal decisions. Consider, for example, these two responses: “what day will you be traveling?” and “what day and time would you like to travel?” If we only consider these responses until the 2nd token (which RMoG does), then the chair might choose the first response due to the absence of a more long-range view of the important token “time” located after the 2nd token. Hence, we also propose a PMoG, which enables the chair to make full use of the prospective predictions of experts as well.

To effectively train MoGNet, we devise a global-and-local (GL) learning scheme. The local loss is defined on a segment of data with a certain intent, which forces each expert to specialize. The global loss is defined on all data, which forces the chair and all experts to coordinate with each other. The global loss can also improve data utilization by enabling the backpropagation error of each data sample to influence all experts as well as the chair.

To verify the effectiveness of MoGNet, we conduct experiments on the MultiWOZ benchmark dataset. MoGNet significantly outperforms state-of-the-art DRG methods, improving over the best performing model on this dataset by 5.64% in terms of overall performance (0.5*inform + 0.5*Success + BLEU) and 0.97% in terms of response generation quality (Perplexity).

The main contributions of this paper are:

- a novel MoGNet model that is the first framework that devises chair and expert generators for DRG, to the best of our knowledge;
- two novel coordination mechanisms, i.e., RMoG and PMoG, to help the chair make better decisions; and
- a GL learning scheme to differentiate experts and fuse data efficiently.

2 MIXTURE-OF-GENERATORS NETWORK

We focus on task-oriented DRG (a.k.a. the context-to-text generation task [3]). Formally, given a current dialogue context \( X = (U, B, D) \), where \( U \) is a combination of previous utterances, \( B \) are the belief states, and \( D \) are the retrieved database results based on \( B \), the goal of task-oriented DRG is to generate a fluent natural language response \( Y = (y_1, \ldots, y_n) \) that contains appropriate system actions to help users accomplish their task goals, e.g., booking a flight ticket.

We propose MoGNet to model the generation probability \( P(Y | X) \).

2.1 Overview

The MoGNet framework consists of two types of roles:

- \( k \) expert generators, each of which is specialized for a particular intent, e.g., a domain, a type of action of a system, etc. Let \( D = \{(X_p, Y_p)\}_{p=1}^{|D|} \) denote a dataset with \( |D| \) independent samples of \((X, Y)\). Expert-related intents partition \( D \) into \( k \) parts \( S = \{S_i\}_{i=1}^k \), where \( S_i = \{(X_p, Y_p)\}_{p=1}^{|S_i|} \). Then \( S_i \) is used to train each expert by predicting \( P(Y | X^i) \). We expect the \( i \)-th expert to perform better than the others on \( S_i \).

- a chair generator, which learns to coordinate a group of experts to make an optimal decision. The chair is trained to predict \( P(Y | X) \), where \( (X, Y) \) is a sample from \( D \).

Fig. 2 shows our implementation of MoGNet; it consists of three types of components, i.e., a shared context encoder, \( k \) expert decoders, and a chair decoder.

2.2 Shared context encoder

The role of the shared context encoder is to read the dialogue context \( X \) and construct a representation. We follow Budzianowksi et al. [3] and model the current dialogue context as a combination of user utterances \( U \), belief states \( B \), and retrieval results from a database \( D \).

First, we employ a Recurrent Neural Network (RNN) \( f \) to map a sequence of input tokens \( U = \{w_1, \ldots, w_m\} \) to hidden vectors.
H^U = \{h^U_1, \ldots, h^U_n\}. The hidden vector h_i at the i-th step can be represented as:

\[ h_{i}^U, s_i = \text{RNN}(w_i, h^U_{i-1}, s_{i-1}), \]

where w_i is the embedding of the token w_i. The initial state s_0 of the RNN is set to 0.

Then, we represent the current dialogue context x as a combination of the user utterance representation h^U_n, the belief state vector h^B, and the database vector h^D:

\[ x = \tanh(W_n h^U_n + W_h h^B + W_d h^D), \]

where h^U_n is the final hidden state from Eq. 1; h^B is a 0-1 vector with each dimension representing a state (slot-value pair); h^D is also a 0-1 vector, which is built by querying the database with the current state B. Each dimension of h^D represents a particular result from the database (e.g., whether a flight ticket is available).

### 2.3 Expert decoder

Given the current dialogue context X and the current decoded tokens Y_{0:j-1}, the l-th expert outputs the probability \( P(y^l_j \mid Y_{0:j-1}, X) \) over the vocabulary \( \mathcal{V} \) at the j-th step by:

\[
P(y^l_j \mid Y_{0:j-1}, X) = \text{softmax}(U^F o^l_j + b),
\]

\[
o^l_j, s^l_j = \text{RNN}(y_{j-1} \oplus c^l_j, o^l_{j-1}, s^l_{j-1}),
\]

where U is the parameter matrix and b is bias; \( s^l_j \) is the state vector, which is initialized by the dialogue context encoder from the shared context encoder, i.e., \( s^l_0 = x; y_{j-1} \) is the embedding of the generated token at time step \( j-1 \); \( \oplus \) is the concatenation operation; \( c^l_j \) is the context vector which is calculated with a concatenation attention mechanism over the hidden representations from a shared context encoder as follows:

\[ c^l_j = \sum_{i=1}^{n} a^l_{ij} h_i, \]

\[ a^l_{ij} = \frac{\exp(w^l_{ij})}{\sum_{l=1}^{k} \exp(w^l_{ij})}, \]

\[ w^l_{ij} = \alpha^l \tanh(W^l T (h_i \oplus s^l_{i-1}) + b_l), \]

where \( \alpha \) is a set of attention weights; \( \oplus \) is the concatenation operation. \( W_i, b_i, v_i \) are learnable parameters, which are not shared by different experts in our experiments.

### 2.4 Chair decoder

Given the current dialogue context X and the current decoded tokens Y_{0:j-1}, the chair decoder estimates the final token prediction distribution \( P(y_j \mid Y_{0:j-1}, X) \) by combining the prediction probabilities from k experts. Here, we consider two strategies to leverage the prediction probabilities from experts, i.e., RMoG and PMoG. The former only considers expert generator outputs from history (until the (j-1)-th time step), which follows the typical neural Mixture-of-Experts (MoE) architecture [25, 27]. We propose the latter to make the chair generator envision the future (i.e., after the (j-1)-th time step) by exploring expert generator outputs from t extra steps (\( t \in [1, n-j], j \in \mathbb{N} \)). Specifically, the chair determines the prediction \( P(y_j \mid Y_{0:j-1}, X) \) as follows:

\[
P(y_j \mid Y_{0:j-1}, X) = \beta^C_j \cdot P(y^C_j \mid Y_{0:j-1}, X)
\]

\[+ \sum_{l=1}^{k} (\beta^R_{jl} + \beta^P_{jl}) \cdot P(y^l_j \mid Y_{0:j-1}, X),
\]

where \( P(y^C_j \mid Y_{0:j-1}, X) \) is the prediction probability from the chair itself; \( P(y^C_j \mid Y_{0:j-1}, X) \) is the prediction probability from expert \( l; \beta^C_j = [\beta^C_{j1}, \beta^C_{j2}, \ldots, \beta^C_{jk}] \) are normalized coordination coefficients, which are calculated as:

\[
\beta^C_j = \frac{\exp(v^T h_j)}{\sum_{l=1}^{k} \exp(v^T h_j)}
\]

\[
h_j = \text{MLP}(P(y^C_j \mid Y_{0:j-1}, X), h^B_j, h^D_j).
\]

\[ \beta^R_{jl}, \beta^P_{jl} \text{ are estimated w.r.t. } P(y^l_j \mid Y_{0:j-1}, X), \]

\( h^B_j \text{ and } h^D_j \text{, respectively. } h^B_j \text{ is a list of retrospective decoding outputs from all experts, which is defined as follows:}

\[
h^B_j = P(y^l_{1:j-1} \mid y_0, X) + \cdots + P(y^l_{j-1} \mid y_0, X)
\]

\[+ P(y^l_{1:j-1} \mid y_0, X),
\]

where \( y_0 \) is a special token “[BOS]” indicating the start of decoding: \( P(y^l_{1:j-1} \mid y_0, X) \) is the output of expert \( l \) from the 1-st to the (j-1)-th step using Eq. 3; \( h^D_j \) is a list of prospective decoding outputs from all experts, which is defined as follows:

\[
h^D_j = P(y^l_{j:t+1} \mid Y_{0:j-1}, X) + \cdots
\]

\[+ P(y^l_{j:t+1} \mid Y_{0:j-1}, X)
\]

\[+ P(y^l_{j:t+1} \mid Y_{0:j-1}, X),
\]

where \( P(y^l_{j:t+1} \mid Y_{0:j-1}, X) \) are the outputs of expert \( l \) from the j-th to (j + t)-th step. We obtain \( P(y^l_{j:t+1} \mid X) \) by forcing expert \( l \) to generate \( t \) steps using Eq. 3 based on the current generated tokens \( Y_{0:j-1} \).

### 2.5 Learning scheme

We devise a global-and-local learning scheme to train MoGNet. Each expert \( l \) is optimized by a localized expert loss defined on \( S_l \), which forces each expert to specialize on one of the portions of data \( S_l \). We use cross-entropy loss for each expert and the joint loss for all experts is as follows:

\[
\mathcal{L}_{\text{experts}} = \sum_{l=1}^{k} \sum_{(X^l_j, y^l_j) \in S_l} \sum_{j=1}^{n} \mu(y^l_j) \log P(y^l_j \mid Y_{0:j-1}, X),
\]

where \( P(y^l_j \mid Y_{0:j-1}, X) \) is the token prediction by expert \( l \) (Eq. 5) computed on the r-th data sample; \( y^l_j \) is a one-hot vector indicating the ground truth token at \( j \).

We also design a global chair loss to differentiate the losses incurred from different experts. The chair can attribute the source of errors to the expert in charge. For each data sample in \( D \), we calculate the combined taken prediction \( P(y_j \mid Y_{0:j-1}, X) \) (Eq. 3). Then the global loss becomes:

\[
\mathcal{L}_{\text{chair}} = \sum_{r=1}^{|D|} \sum_{j=1}^{n} \log P(y_j \mid Y_{0:j-1}, X).
\]

Our overall optimization follows the joint learning paradigm that is defined as a weighted combination of constituent losses:

\[
\mathcal{L} = \lambda \cdot \mathcal{L}_{\text{experts}} + (1 - \lambda) \cdot \mathcal{L}_{\text{chair}},
\]

where \( \lambda \) is a hyper-parameter to regulate the importance between the experts and the chair for optimizing the loss.
3 EXPERIMENTAL SETUP

3.1 Research questions
We seek to answer the following research questions: (RQ1) Does MoGNet outperform state-of-the-art end-to-end single-module DRG models? (RQ2) How does the choice of a particular coordination mechanism (i.e., RMoG, PMoG, or neither of the two) affect the performance of MoGNet? (RQ3) How does the GL learning scheme compare to using the general global learning as a learning scheme?

3.2 Dataset
Our experiments are conducted on the MultiWOZ [4] dataset. This is the latest large-scale human-to-human TDS dataset with rich semantically labeled, e.g., domains and dialogue actions, and benchmark results. We follow the dominant Sequence-to-Sequence learning algorithm with hyperparameters \( \alpha = 0.005, \beta_1 = 0.9, \beta_2 = 0.999 \) and \( \epsilon = 10^{-8} \). We also apply gradient clipping [22] with range \([-5, 5]\) during training. We use \( l_2 \) regularization to alleviate overfitting, the weight of which is set to \( \epsilon = 10^{-5} \). We set the minibatch size to 64. We use greedy search to generate the responses during testing. Please note that if a data point has multiple intents, then we assign it to each corresponding expert, respectively. The code is available online.

3.3 Model variants and baselines
We consider a number of variants of the proposed mixture-of-generators model:

- **MoGNet**: the proposed model with RMoG and PMoG and GL learning scheme.
- **MoGNet-P**: the model without prospection ability by removing PMoG coordination mechanism from MoGNet.
- **MoGNet-P-R**: the model removing the two coordination mechanisms and remaining GL learning scheme.
- **MoGNet-GL**: the model that removes GL learning scheme from MoGNet.

See Table 1 for a summary. Without further indications, the intents used are based on identifying eight different domains: Attraction, Booking, Hotel, Restaurant, Taxi, Train, General, and UNK.

<table>
<thead>
<tr>
<th>Model</th>
<th>( \beta_1^C )</th>
<th>( \beta_1^R )</th>
<th>( \beta_1^P )</th>
<th>( \lambda )</th>
</tr>
</thead>
<tbody>
<tr>
<td>MoGNet</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>0.5</td>
</tr>
<tr>
<td>MoGNet-P</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>0.5</td>
</tr>
<tr>
<td>MoGNet-P-R</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>0.5</td>
</tr>
<tr>
<td>MoGNet-GL</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>0.0</td>
</tr>
</tbody>
</table>

\( \beta_1^C, \beta_1^R, \beta_1^P \) are from Eq. 5. “True” means we preserve it and learn it as it is. “False” means we remove it (set it to 0). \( \lambda \) is from Eq. 11 and we report two settings, 0.0 and 0.5. See § 2.2.

To answer RQ1, we compare MoGNet with the following methods that have reported results on this task according to the official leaderboard:

- **S2SAttnLSTM**: We follow the dominant Sequence-to-Sequence (Seq2Seq) model under an encoder-decoder architecture [3] and reproduce the benchmark baseline, i.e., single-module model named S2SAttnLSTM [3, 4], based on the source code provided by the authors. See footnote 3.
- **S2SAttnGRU**: A variant of S2SAttnLSTM, with Gated Recurrent Units (GRUs) instead of LSTMs and other settings kept the same.
- **Structured Fusion**: It learns the traditional dialogue modules and then incorporates these pre-trained sequentially dependent modules into end-to-end dialogue models by structured fusion networks [20].
- **LaRLAttnGRU**: The state-of-the-art model [36], which uses reinforcement learning and models system actions as latent variables. LaRLAttnGRU uses ground truth system action annotations and user goals to estimate the rewards for reinforcement learning during training.

3.4 Evaluation metrics
We use the following commonly used evaluation metrics [3, 35]:

- **Inform**: the fraction of responses that provide a correct entity out of all responses.
- **Success**: the fraction of responses that answer all the requested attributes out of all responses.
- **BLEU**: for comparing the overlap between a generated response to one or more reference responses.
- **Score**: defined as \( \text{Score} = (0.5 \times \text{Inform} + 0.5 \times \text{Success} + \text{BLEU}) \times 100 \). This measures the overall performance in term of both task completion and response fluency [20].
- **PPL**: denotes the perplexity of the generated responses, which is defined as the exponentiation of the entropy. This measures how well a probability DRG model predicts a token in a response generation process.

We use the toolkit released by Budzianowski et al. [3] to compute the metrics following their settings, we also use Score as the selection criterion to choose the best model on the validation set and report the performance of the model on the test set. We use a paired t-test to measure statistical significance \( (p < 0.01) \) of relative improvements.

3.5 Implementation details
Theoretically, the training time complexity of each data sample is \( \mathcal{O}(n \times (k + 1) \times n) \), where \( n \) is the number of response tokens. To reduce the computation cost, we assign \( j + t = n \) and compute the expert prediction with Eq. 3. This means that the chair will make a final decision only after all the experts have decoded their final tokens. Thus, the time complexity decreases to \( \mathcal{O}(n \times (k + 1) \times n) \).

For a fair comparison, the vocabulary size is the same as Budzianowski et al. [4], which has 400 tokens. Out-of-vocabulary words are replaced with “[UNK]”. We set the word embedding size to 50 and all GRU hidden state sizes to 150. We use Adam [13] as our optimization algorithm with hyperparameters \( \alpha = 0.005, \beta_1 = 0.9, \beta_2 = 0.999 \) and \( \epsilon = 10^{-8} \). We also apply gradient clipping [22] with range \([-5, 5]\) during training. We use \( l_2 \) regularization to alleviate overfitting, the weight of which is set to \( 10^{-5} \). We set the minibatch size to 64. We use greedy search to generate the responses during testing. Please note that if a data point has multiple intents, then we assign it to each corresponding expert, respectively. The code is available online.

4 RESULTS

4.1 Automatic evaluation
We evaluate the overall performance of MoGNet and the comparable baselines on the metrics defined in § 3.4. The results are shown in Table 2. First of all, MoGNet outperforms all baselines by a large margin in terms of overall performance metric, i.e., satisfaction Score. It significantly outperforms the state-of-the-art baseline LaRLAttnGRU by 5.64% (Score) and 0.97 (PPL). Thus, MoGNet
not only improves the satisfaction of responses but also improves the quality of the language modeling process. MoGNet also achieves more than 6.70% overall improvement over the benchmark baseline S2SAttnLSTM and its variant S2SAttnGRU. This proves the effectiveness of the proposed MoGNet model.

Second, LaRLAttnGRU achieves the highest performance in terms of Success, followed by MoGNet. However, it results in a 7.33% decrease in BLEU and a 2.56% decrease in Informativeness compared to MoGNet. Hence, LaRLAttnGRU is good at answering all requested attributes but not as good at providing more appropriate entities with high fluency as MoGNet. LaRLAttnGRU tends to generate more slot values to increase the probability of answering the requested attributes. Take an extreme case as an example: if we force a model to generate all tokens with slot values, then it will achieve an extremely high Success but a low BLEU.

Third, S2SAttnLSTM is the worst model in terms of overall performance (Score). But it achieves the best PPL. It tends to generate frequent tokens from the vocabulary which exhibits better language modeling characteristics. However, it fails to provide useful information (the requested attributes) to meet the user goals. By contrast, MoGNet improves the user satisfaction (i.e., Score) greatly and achieves response fluency by taking specialized generations from all experts into account.

4.2 Human evaluation

To further understand the results in Table 2, we conducted a human evaluation of the generated responses from S2SAttnGRU, LaRLAttnGRU, and MoGNet. We ask workers on Amazon Mechanical Turk (AMT) to read the dialogue context, and choose the responses that satisfy the following criteria: (i) Informativeness measures whether the response provides appropriate information that is requested by the user query. No extra inappropriate information is provided. (ii) Consistency measures whether the generated response is semantically aligned with the ground truth response. (iii) Satisfaction measures whether the response has a satisfactory performance promising both Informativeness and Consistency. As with existing studies [20], we sample one hundred context-response pairs to do human evaluation. Each sample is labeled by three workers. The workers are asked to choose either all responses that satisfy the specific criteria or the “NONE” option, which denotes none of the responses satisfy the criteria. To make sure that the annotations are of high quality, we calculate the fraction of the responses that satisfy each criterion out of all responses that passes the golden test. That is, we only consider the data from the workers who have chosen the golden response as an answer.

The results are displayed in Table 3. MoGNet performs better than S2SAttnGRU and LaRLAttnGRU on Informativeness because it frequently outputs responses that provide richer information (compared with S2SAttnGRU) and fewer extra inappropriate information (compared with LaRLAttnGRU). MoGNet obtains the best results, which means MoGNet is able to generate responses that are semantically similar to the golden responses with large overlaps. The results of LaRLAttnGRU outperforms S2SAttnGRU in all cases except for Satisfaction under the strict condition (>). This reveals that balancing between Informativeness and Consistency makes it difficult for the mturk workers to assess the overall quality measured by Satisfaction. In this case, MoGNet receives the most votes on Satisfaction under the strict condition (>) as well as the loose condition (≥). This shows that the workers consider the responses from MoGNet more appropriate than the other two models with a high degree of agreement. To sum up, MoGNet is able to generate user-favored responses in addition to the improvements for automatic metrics.

4.3 Coordination mechanisms

In Table 2 we contrast the effectiveness of different coordination mechanisms. We can see that MoGNet-P loses 4.32% overall performance with a 0.62% decrease of BLEU, 5.90% decrease of Informativeness and 1.50% decrease of Success. This shows that the prosecution design of the PMoG mechanism is beneficial to both task completion and response fluency. Especially, most improvements come from providing more correct entities while improving generation fluency. MoGNet-P-R reduces 2.62% Score with 1.97% lower of BLEU, 0.2% lower of Informativeness and 1.10% of Success. Thus, the MoGNet framework is effective thanks to its design with two types of roles: the chair and the experts.

4.4 Learning scheme

We use MoGNet-GL to refer to the model that removes the GL learning scheme from MoGNet and uses the general global learning instead. MoGNet-GL results in a sharp reduction of 6.95% overall performance with 0.80% of BLEU, 6.90% of Informativeness and 5.40% of Success. The main improvement is attributed to the strong task completion ability. This shows the effectiveness and importance of the GL learning scheme as it encourages each expert to specialize on a particular intent while the chair prompts all experts to coordinate with each other.

5 ANALYSIS

In this section, we explore MoGNet in more detail. In particular, we examine (i) whether the intent partition affects the performance of MoGNet ([52]), (ii) whether the improvements of MoGNet could simply be attributed to having a larger number of parameters ([52]), (iii) how the hyper-parameter λ (Eq. [1]) affects the performance of MoGNet.
5.1 Intent partition analysis

As stated above, the responses vary a lot for different intents which are differentiated by the domain and the type of system action. Therefore, we experiment with two types of intents as shown in Table 6.

Table 6: Two groups of intents that are divided by domains and the type of system actions.

<table>
<thead>
<tr>
<th>Type</th>
<th>Intents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain</td>
<td>Attraction, Booking, Hotel, Restaurant, Taxi, Train, General, UNK.</td>
</tr>
<tr>
<td>Action</td>
<td>Book, Inform, NoBook, Offer, OfferBook, OfferBooked, Select, Recommend, Request, Bye, Greet, Reqmore, Welcome, UNK.</td>
</tr>
</tbody>
</table>

MoGNet consistently outperforms the baseline S2SAttnGRU for both ways of partitioning intents. Interestingly, MoGNet-domain greatly outperforms MoGNet-action. We believe there are two reasons: First, the system actions are not suitable for grouping intents because some partition subsets are hard to be distinguished from each other, e.g., OfferBook and OfferBooked. Second, some system actions only have a few data samples, simply not enough to specialize the experts. The results show that different ways of partitioning intents may greatly affect the performance of MoGNet. Therefore, more effective intent partition methods, e.g., adaptive implicit intent partitions, need to be explored in future work.

5.2 Hyper-parameter analysis

To address (ii), we show the results of MoGNet and S2SAttnGRU with different hidden sizes in Fig. 3. S2SAttnGRU outperforms MoGNet when the number of parameters is less than 0.6e7. However, MoGNet achieves much better results with more parameters. Most importantly, the results from both models show that a larger number of parameters is important for searching a train ticket (1st turn) and omit the exact departure time ([value_time]) of the train (2nd turn). Without GL, MoGNet-GL ignores the primary time information need day (1st turn) and omits the implicit need of [value_price] (4th turn). There are also some low-quality cases, e.g., MoGNet and the baselines occasionally generate redundant and lengthy responses, because none of them has addressed this issue explicitly during training.

6 RELATED WORK

Traditional models for DRG [8,33] decompose the task into sequentially dependent modules, e.g., Dialogue State Tracking (DST) [27], Policy Learning (PL) [35], and Natural Language Generation (NLG) [21]. Such models allow for targeted failure analyses, but inevitably incur upstream propagation problems [8]. Recent work views DRG as a source-to-target transduction problem, which maps a dialogue context to a response [11,17,31]. Sordoni et al. [28] show that using an RNN to generate text conditioned on dialogue history results in more natural conversations. Later improvements include the addition of attention mechanisms [16,29], modeling the hierarchical structure of dialogues [26], or jointly learning belief and action values of [value] [11,17,31].
Table 8: Example responses from MoGNet with the ablation settings in a 4-turn dialogue.

<table>
<thead>
<tr>
<th>Turn</th>
<th>User</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>i am looking for a train departing from [value_place] that will arrive at [value_place].</td>
<td>S2SAttnGRU: what day would you like to travel? LaRLAttnGRU: there are [value_count] trains leaving [value_place] on [value_day]. what day would you like to travel?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MoGNet: what day and time would you like to travel? MoGNet-P: what day will you be traveling? MoGNet-P-R: what day will you be traveling? MoGNet-GL: there are [value_count] trains... do you have a time you would like to depart?</td>
</tr>
<tr>
<td>2</td>
<td>i would like to leave on [value_day] and arrive no later than [value_time].</td>
<td>S2SAttnGRU: [train_id] will arrive at [value_time]. LaRLAttnGRU: i have train [train_id] leaving at [value_time] and arriving at [value_time]. would you like me to book this for you?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MoGNet: [train_id] leaves at [value_time] and arrives by [value_time]. MoGNet-P: [train_id] will arrive at [value_time]. MoGNet-P-R: the [train_id] arrives at [value_time]. MoGNet-GL: i have train [train_id] that leaves at [value_time] and arrives at [value_time]. +All: would you like me to book that for you?</td>
</tr>
<tr>
<td>3</td>
<td>can i please have the reference number? All: how many tickets do you need?</td>
<td>S2SAttnGRU: i have booked you [value_count] tickets on the [train_id]. your reference number is [train_reference]. is there anything else i can help you with? LaRLAttnGRU: i have booked [value_count] tickets on [train_id], the reference number is [train_reference].</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MoGNet: i have booked [value_count] tickets for you. the total fee is [value_price] gbp payable at the station. your reference number is [train_reference]. is there anything else i can help you with? MoGNet-P: i have booked [value_count] tickets. the total fee is [value_price] gbp payable at the station. your reference number is [train_reference]. MoGNet-P-R: booking was successful, the total fee is [value_price] gbp payable at the station. reference number is [train_reference]. MoGNet-GL: i have booked [value_count] tickets for you. the reference number is [train_reference]. is there anything else i can help you with?</td>
</tr>
<tr>
<td>4</td>
<td>could you book [value_count] tickets for me and send me the reference number?</td>
<td>All: would you like me to book that for you?</td>
</tr>
</tbody>
</table>
REFERENCES


