Cross-Domain Slot Filling as Machine Reading Comprehension

Mengshi Yu*, Jian Liu†, Yufeng Chen, Jinan Xu† and Yujie Zhang
Beijing Jiaotong University, Beijing, China
{19120432, jianliu, chenyf, jaxu, yjzhang}@bjtu.edu.cn

Abstract

With task-oriented dialogue systems being widely applied in everyday life, slot filling, the essential component of task-oriented dialogue systems, is required to be quickly adapted to new domains that contain domain-specific slots with few or no training data. Previous methods for slot filling usually adopt sequence labeling framework, which, however, often has limited ability when dealing with the domain-specific slots. In this paper, we take a new perspective on cross-domain slot filling by framing it as a machine reading comprehension (MRC) problem. Our approach firstly transforms slot names into well-designed queries, which contain rich informative prior knowledge and are very helpful for the detection of domain-specific slots. In addition, we utilize the large-scale MRC dataset for pre-training, which further alleviates the data scarcity problem. Experimental results on SNIPS and ATIS datasets show that our approach consistently outperforms the existing state-of-the-art methods by a large margin.

1 Introduction

Building a task-oriented dialogue system that can comprehend users’ requests and satisfy their needs has been a key component in many intelligent conversation applications [Jaech et al., 2016; Gao et al., 2020; Liang et al., 2020]. As an indispensable part of task-oriented dialogue systems, slot filling aims to identify task-related slot types in certain domains. For instance, as shown in Figure 1, given the user request “book the hat for my classmates” in domain BookRestaurant, we need to fill domain-specific roles like “restaurant_name” and “party_size_description” with “the hat” and “my classmates”, respectively. Previous methods for slot filling often focus on supervised learning [Zhang and Wang, 2016; Goo et al., 2018; Wu et al., 2020], where large-scale labeled datasets are required. However, slot filling faces the rapid changing of domains, and few or no target training data may be available in a new domain. To alleviate the data scarcity problem in target domains, we need to train a model that can borrow the prior experience from source domains and adapt it to target domains quickly with limited training samples.

Conventional approaches [Zhang and Wang, 2016; Goo et al., 2018; Wu et al., 2020] take slot filling as a sequence labeling task, which assigns a label to each token in a given sequence, as shown in Figure 1. However, the sequence labeling framework is data-hungry and does not have the potential to scale to new domains that consist of domain-specific slots and usually have few or no training data. To address these issues, [Shah et al., 2019; Liu et al., 2020b; He et al., 2020] add meta-information such as slot descriptions and slot examples to capture the semantic relationship between slot types and input tokens. However, these methods also require slot definitions to be similar between training data and unseen test data. That is, if such systems face completely new slot types (unseen slots), their performances would degrade significantly (As seen in our experiments of unseen slots in subsection 4.2).

In this paper, we propose a new approach for cross-domain slot filling, which frames the task as a machine reading com-

User Request: “book the hat for my classmates”

1) Slot filling via sequence labeling

<table>
<thead>
<tr>
<th>Book</th>
<th>O</th>
<th>For</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>The</td>
<td>B-r_n</td>
<td>My</td>
<td>B-p_s_d</td>
</tr>
<tr>
<td>Hat</td>
<td>I-r_n</td>
<td>Classmates</td>
<td>I-p_s_d</td>
</tr>
</tbody>
</table>

2) Slot filling as reading comprehension

\[ Q_{i,r} \text{: what is the name of the restaurant?} \quad A_{i,r} \text{: “the hat”} \]

\[ Q_{i,p,s,d} \text{: who are the people attending the party?} \quad A_{i,p,s,d} \text{: “my classmates”} \]

Figure 1: An example of slot filling via sequence labeling framework and reading comprehension framework. In the sequence labeling framework, slot labels are annotated in “BIO” format: “B” represents the start of a slot span, “I” the inside of a span while “O” denotes that the word does not belong to any slot. In the reading comprehension framework, each slot type corresponds to a well-designed question \( Q_i \) where \( i \) denotes the \( i \)-th slot we need to fill, and we use answer \( A_i \) of the question \( Q_i \) to fill the \( i \)-th slot. \( “r_n” \) and \( “p_s, d” \) are short for slot names “restaurant_name” and “party_size_description”, respectively.

*Equal Contribution
†Corresponding Author
Code and data available at https://github.com/mengshiY/RCSF
prehension (MRC) problem [Hermann et al., 2015]. An example of slot filling in the MRC framework is illustrated in Figure 1. We transform each slot type we need to fill into a question, and then fill the slot by answering the question. Specifically, we design three strategies to generate the questions, which will be discussed in detail later. After the questions are generated, we build a BERT-based MRC model [Devlin et al., 2019] to answer each of the questions and synthesize the answers as the final results. In order to boost slot filling in low-resource scenarios, we also leverage the large-scale MRC dataset for pre-training. Compared with the traditional sequence labeling framework, MRC framework has the advantage of introducing prior knowledge about slot information into the queries. More importantly, by converting the sequence labeling problem into MRC problem, we can make full use of large-scale MRC datasets to learn semantic information, which is beneficial for slot filling tasks in the cross-domain setting.

To verify the effectiveness of our approach, we conduct extensive experiments on two benchmark datasets. For SNIPS dataset, our approach achieves performance gains over current state-of-the-art model by 18.37%, 21.28% and 15.43%, respectively under zero-shot setting, 20-shot setting and 50-shot setting. For ATIS dataset, our approach outperforms the existing state-of-the-art by 27.78% under zero-shot setting, 25.69% under 20-shot setting and 18.79% under 50-shot setting. Moreover, further experiments show that even without the model pre-training, our model still achieves better results consistently than the existing state-of-the-art approaches. In addition, we also investigate the effect of different query generation strategies and find that adding high-quality slot examples into the queries can further improve the model performance under zero-shot setting.

Our main contributions can be summarized as follows:

- We propose a new MRC framework to deal with cross-domain slot filling. Compared with previous sequence labeling approaches, our framework can introduce more prior knowledge into the well-designed queries, and thus improve its performance in zero-shot setting. Moreover, by converting slot filling task into MRC task, we are able to utilize the large-scale supervised MRC dataset for pre-training and further improve the performance.
- We devise different strategies to transfer a slot into a query, and conduct a series of studies to explore their effects.
- We conduct extensive experiments on two commonly used datasets and show that our approach consistently outperforms the existing state-of-the-art model by a large margin.

## 2 Related Work

### 2.1 Cross-Domain Slot Filling

There are mainly two challenges in cross-domain slot filling task. One is to adapt the shared slot types from source domains to target domains, and the other is to handle domain-specific slot types which have few or no supervision signals for training.

To deal with the shared slot types, a common approach is transfer learning (TF). TF aims to adapt the learned source model $M_S$ trained on source domain $D_S$ to produce a target model $M_T$ for target domain $D_T$. TF can be categorized into data-driven transfer and model-driven transfer. Data-driven transfer approaches are based on pre-training and fine-tuning mechanisms. [Goyal et al., 2018] train $M_S$ on large-scale $D_S$, and then fine-tune $M_S$ by replacing the output layer corresponding with the label space from $D_T$ and further train the model on $D_T$. [Siddhant et al., 2019] leverage large-scale unlabeled data to learn contextual embedding, i.e., ELMo [Peters et al., 2018], before fine-tuning on $D_T$. Different from data-driven approaches, model-driven [Kim et al., 2017; Jha et al., 2018] approaches alleviate the slot adaptation problem by enabling model re-usability. Although different domains have different slot types, common slots such as “date”, “time” and “country” can be shared. These approaches usually first train $M_S$ on these reusable slots, and then the outputs of $M_S$ are used to guide the training of $M_T$ for new slots.

While TF approaches can share knowledge learned on different domains, such models can not handle unseen slots. Therefore, researchers [Bapna et al., 2017; Guerini et al., 2018; Lee and Jha, 2019; Shah et al., 2019; Liu et al., 2020b] start to investigate zero-shot methods, which can be broadly classified into two categories. One is to train the model on slot descriptions which carry information about the slots [Bapna et al., 2017; Lee and Jha, 2019; Liu et al., 2020b]. Slots with similar meanings would have similar descriptions, so it is possible to recognize the unseen slots by training on similar seen slots. The other zero-shot approach explores the usage of slot examples [Shah et al., 2019; Guerini et al., 2018], showing that using a small number of slot examples along with slot descriptions performs better than using the slot descriptions alone. However, these zero-shot approaches simply use the slot information to match its most corresponding entities and require slot information to be similar between the seen slots and the unseen slots, which limits their performance. Unlike these work, we utilize the slot information in a more natural way, that is, we transform it into natural questions and get slot entities by answering the questions.

### 2.2 Framing Other NLP Tasks as MRC

Machine Reading Comprehension models [Hermann et al., 2015] predict answer spans from a context through a given query. Recently, there has been a trend of transforming NLP tasks to MRC problems. For example, [McCann et al., 2018] use the MRC framework to implement ten different NLP tasks uniformly and all achieve competitive performances. [Li et al., 2020] transform named entity recognition (NER) task into MRC to handle the nested NER problem. [Gao et al., 2020] leverage MRC datasets and use MRC techniques to enhance dialogue state tracking task. [Liu et al., 2020a] propose an unsupervised question generation method and utilize a BERT-based question-answering process to bridge MRC and event extraction problem.

Inspired by the great success of MRC, we exploit it to deal with cross-domain slot filling task. To the best of our knowledge, there is currently no specific research for cross-domain...
3 Methodology

Our approach, denoted by RCSF (Reading Comprehension for Slot Filling) is depicted in Figure 2. Given a user request, we are supposed to fill its corresponding slots with tokens in it. First of all, RCSF generates a query for each slot with different strategies. Then, the query and the user request are concatenated together as the inputs of the RCSF model (we use BERT as the backbone in this paper). RCSF predicts the start and end indexes based on the hidden representation generated by BERT and matches the start indexes and end indexes as the answer spans through the start-end matching module. Finally, RCSF fills the slots with answers of their related queries.

3.1 Task Formulation

Given a user request $X = \{x_1, x_2, \cdots, x_n\}$ with $n$ words and a predefined set of slot types $S_Y$ in domain $D$, we need to fill each slot type $y \in S_Y$ with entities in $X$. We convert the tagging-style annotated slot filling dataset to a set of $(query, context, answer)$ triples. For each slot type $y \in S_Y$, it is associated with a natural language question (query) $q_y = \{q_{y1}, q_{y2}, \cdots, q_{ym}\}$ where $m$ denotes the length of the generated question. So slot tagging can be transformed to predicting the answer spans of the specific slot $y = [(s_1, e_1), (s_2, e_2), \cdots, (s_t, e_t)]$ where $s_t$ and $e_t$ denotes the start and the end position of the $i$-th span, respectively, and $t$ is the number of spans ($1 \leq i \leq t$).

3.2 Query Generation

In our MRC framework, we firstly transform each slot type into its corresponding query which contains prior knowledge we need. The strategy to generate queries is important for cross-domain slot filling, especially in zero-shot setting. Since the BERT model we use is pre-trained on the MRC dataset in which queries are natural questions, we are supposed to generate natural questions as well to utilize the semantic information of the pre-trained model. Therefore, we use templates such as “what is the _?”, where the blank is filled with slot information, to construct queries. As the examples shown in Table 1, we propose the following three strategies for query generation:

- **Description**: We turn slot names into their corresponding slot descriptions by replacing punctuation marks like “),” and “,” with blanks and replacing abbreviations with their original words. Queries are constructed by filling the above template using the slot descriptions directly.
- **Back-translation**: As above, we firstly use the slot descriptions to construct questions. However, the simple conjunction of the template and slot descriptions may introduce extra noises caused by grammar errors. To wipe off the noises, we translate the constructed questions into Chinese and re-translate them back into English. After the round-trip translation, the queries are more natural.
- **Example**: The queries are constructed using slot de-
3.3 Slot Filling as Answer Prediction

Given the query \( q_g \) and the context \( X \), we need to extract the answer spans \( z_g \) under the MRC framework. BERT [Devlin et al., 2019] is used as the backbone. As depicted in Figure 2, we concatenate the question \( q_g \) and the input sentence \( X \) as the input sequence \( I = [CLS], q_1, q_2, \ldots, q_m, [SEP], x_1, x_2, \ldots, x_n, [SEP] \) to BERT where \([CLS] \) and \([SEP] \) stand for the classifier token and sentence separator token in BERT, respectively. Then BERT receives the input sequence and generates a context representation matrix \( H \subseteq \mathbb{R}^{n \times d} \), where \( d \) is the dimension of the last layer of BERT.

Start and End Prediction

In the traditional MRC framework, one query usually has one answer. However, in our approach, one query may correspond to multiple answers. Therefore, we construct two binary classifiers. One is used to predict whether the token is a start index, and the other is employed to predict whether the token is an end index. Given the representation matrix \( H \) output by BERT, the model first predicts the probability \( P_{start} \) of each token being a start index as follows:

\[
L_{start} = \text{Linear}(HW_{start}), \quad L_{start} \subseteq \mathbb{R}^{n \times 2} \tag{1}
\]

\[
P_{start} = \text{Softmax}(L_{start}V_{start}), \quad P_{start} \subseteq \mathbb{R}^{n \times 2} \tag{2}
\]

where \( \text{Linear} \) denotes a fully connected layer and \( \text{Softmax} \) represents the softmax function. \( W_{start} \) and \( V_{start} \) are trainable weights.

And the end index prediction procedure is exactly the same, except that we have other matrix \( W_{end} \) and \( V_{end} \) to obtain the probability matrix \( P_{end} \) of each token being an end index:

\[
L_{end} = \text{Linear}(HW_{end}), \quad L_{end} \subseteq \mathbb{R}^{n \times 2} \tag{3}
\]

\[
P_{end} = \text{Softmax}(L_{end}V_{end}), \quad P_{end} \subseteq \mathbb{R}^{n \times 2} \tag{4}
\]

Start-End Matching

In the context \( X \), there can be multiple entities of the same category, which means we are supposed to predict multiple start-end pairs. Traditional methods [Sun et al., 2020] get the start-end pairs by matching the start index with its nearest end index, which does not work well here since the predicted slot entities could overlap and we might lose the most possible one when eliminating overlaps. So we adopt the principle of the most possible pair first. That is, we first sort the start indexes and end indexes by their probability \( P_{start} \) and \( P_{end} \). Then we choose the top-\( N \) start indexes and the top-\( N \) end indexes, where \( N \) is a predefined number:

\[
I_{start} = \{i|P_{start}^i > t, i = 1, 2, \ldots, N\} \tag{5}
\]

\[
I_{end} = \{j|P_{end}^j > t, j = 1, 2, \ldots, N\} \tag{6}
\]

where \( i \) and \( j \) denotes the \( i \)-th and \( j \)-th rows of a matrix respectively and \( t \) is the minimum probability of the top-\( N \) indexes.

With the sets of the most possible start indexes \( I_{start} \) and end indexes \( I_{end} \), we calculate the probability \( P_{ij} \) of each start-end pair by adding \( P_{start}^i \) and \( P_{end}^j \) where \( P_{start}^i \) denotes the probability of the \( i \)-th token being a start token and \( P_{end}^j \) denotes the \( j \)-th token being an end token. Then, we sort the matched start-end pairs by \( P_{ij} \) and choose the most possible pair which does not overlap the chosen ones.

3.4 Train and Test

To utilize the large-scale MRC dataset, we adopt a two-stage training procedure. Our MRC model is first pre-trained on the MRC dataset SQuAD2.0 [Rajpurkar et al., 2018], and then fine-tuned on queries and answers created from our slot filling datasets.

In the training stage, each context \( X \) is paired with two label sequences \( Y_{start} \) and \( Y_{end} \), which denote the ground-truth label of each token \( x_i \) being the start index and the end index of an entity respectively. The loss functions are defined as follows:

\[
\text{Loss}_{start} = CE(P_{start}, Y_{start}) \tag{7}
\]

\[
\text{Loss}_{end} = CE(P_{end}, Y_{end}) \tag{8}
\]

\[
\text{Loss} = \lambda \text{Loss}_{start} + (1 - \lambda) \text{Loss}_{end} \tag{9}
\]

where \( CE \) represents the cross-entropy loss function and \( \lambda \in [0, 1] \) is a balanced factor used to control the overall training objectives. In our experiment, we set \( \lambda = 0.5 \) according to the performance of the model on the validation set.

At test time, first of all, start and end indexes are separately selected based on Eq. 5 and Eq. 6. Then the start-end matching module is applied to align the extracted start indexes and the end indexes, leading to the final extracted answers.

4 Experiments

4.1 Experimental Settings

Datasets

We evaluate our framework on SNIPS [Coucke et al., 2018], a public spoken language understanding dataset which contains 39 slot types across seven domains (intents) and about 2000 samples per domain. To simulate the cross-domain scenarios, we follow [Liu et al., 2020b] to split the dataset, that is, we choose one domain as the target domain and the other six domains as the source domains each time.

However, domains in SNIPS are not completely independent with each other. We use another commonly used dataset ATIS [Hemphill et al., 1990] as target domain to test our model. It consists of 5971 utterances related to the airline travel domain with 83 slot types.

Baselines

We compare our approach with the following baselines:

- **Concept Tagger (CT):** A method proposed by [Bapna et al., 2017], which utilizes slot descriptions to boost the performance on detecting unseen slots.

- **Robust Zero-shot Tagger (RZT):** Based on CT, [Shah et al., 2019] leverage both slot descriptions and examples to improve the robustness of zero-shot slot filling.
Table 2: F1-scores (%) on SNIPS (SN) and ATIS (AT) for different target domains under zero-shot and few-shot learning settings. Scores in each row represents the performance of the leftmost domain, and RCSF denotes our approach using queries constructed by slot descriptions. Since the SNIPS dataset consists of multiple domains, we calculate the average F1 of all domains. We also try Coach with BERT encoders, but it did not perform better than its original LSTM encoders.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Training Setting</th>
<th>Zero-shot</th>
<th>Few-shot on 20 (1%) samples</th>
<th>Few-shot on 50 (2.5%) samples</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Domain Model →</td>
<td>CT RZT Coach</td>
<td>RCSI CT RZT Coach</td>
<td>RCSI CT RZT Coach</td>
</tr>
<tr>
<td>SN</td>
<td>AddToPlaylist</td>
<td>38.82 42.77 50.90</td>
<td>68.70 58.36 63.18 62.76</td>
<td>88.37 68.69 74.89 74.68</td>
</tr>
<tr>
<td></td>
<td>BookRestaurant</td>
<td>27.54 30.68 34.01</td>
<td>63.49 45.65 50.54 65.97</td>
<td>85.56 54.22 54.49 74.82</td>
</tr>
<tr>
<td></td>
<td>GetWeather</td>
<td>46.45 30.28 50.47</td>
<td>65.36 54.22 58.86 67.89</td>
<td>88.83 63.23 58.87 79.64</td>
</tr>
<tr>
<td></td>
<td>PlayMusic</td>
<td>32.86 33.12 32.01</td>
<td>53.51 46.35 47.20 54.04</td>
<td>80.95 54.32 59.20 66.38</td>
</tr>
<tr>
<td></td>
<td>RateBook</td>
<td>14.54 16.43 22.06</td>
<td>36.51 64.37 63.33 74.68</td>
<td>93.35 76.45 75.87 84.62</td>
</tr>
<tr>
<td></td>
<td>SearchCreativeWork</td>
<td>39.79 44.45 46.65</td>
<td>69.22 57.83 63.39 57.19</td>
<td>81.30 66.38 67.81 64.56</td>
</tr>
<tr>
<td></td>
<td>SearchScreeningEvent</td>
<td>13.83 12.25 25.63</td>
<td>33.54 48.59 49.18 67.38</td>
<td>80.46 70.67 74.58 83.85</td>
</tr>
<tr>
<td></td>
<td>Average F1</td>
<td>30.55 32.85 37.39</td>
<td>55.76 53.62 56.53 64.27</td>
<td>85.55 64.85 66.67 75.51</td>
</tr>
<tr>
<td>AT</td>
<td>AirlineTravel</td>
<td>2.14 2.86 1.64</td>
<td>30.64 26.05 41.37 54.91</td>
<td>80.60 35.87 51.80 66.99</td>
</tr>
</tbody>
</table>

Table 3: Averaged F1-scores (%) over all target domains on SNIPS (SN) for unseen slots and unseen sentences. Scores in each row represent the performance of the leftmost method, and RCSF denotes our approach using queries constructed by slot descriptions. Table 2 demonstrates the main results of our MRC model compared to the baselines. In SNIPS dataset, our approach outperforms the state-of-the-art model (Coach) by 18.37% on the average F1 under zero-shot setting, 21.28% under 20-shot setting and 15.43% under 50-shot setting, which demonstrates the effectiveness of our method. To simulate the cross-domain situation in real world, we also test our model on ATIS dataset with SNIPS dataset as the training set. In this setting, AirlineTravel in ATIS is considered as the target domain while all of the seven domains in SNIPS are taken as the source domains. Our model still outperforms the existing coarse-grained slot type prediction. Slot descriptions are used in the second stage to help recognize unseen slots, and template regularization is applied to further improve the slot filling performance of similar or the same slot types.

### Implementation Details

We conduct our experiment based on BertForQuestionAnswering3 implemented by HuggingFace as our base model, and load the pre-trained weights provided by deepset4. They pre-train the BERT model on the question answering dataset SQuAD2.0 [Rajpurkar et al., 2018]. Adam optimizer [Kingma and Ba, 2014] is applied to optimize all parameters with a learning rate 1e-5. We set the batch size to 64 and the maximum sequence length to 128. The patience of early stop is set to 5. As for the baseline models, we use the implementation of [Liu et al., 2020b] and follow the same settings for a fair comparison. F1-score is used as the evaluation metric. A slot span is considered to be correct only if its range and slot type are both correct. We fine-tune all hyper-parameters on the validation set and use the best checkpoint to test our model.

### 4.2 Main Results and Discussions

Table 2 demonstrates the main results of our MRC model compared to the baselines. In SNIPS dataset, our approach outperforms the state-of-the-art model (Coach) by 18.37% on the average F1 under zero-shot setting, 21.28% under 20-shot setting and 15.43% under 50-shot setting, which demonstrates the effectiveness of our method. To simulate the cross-domain situation in real world, we also test our model on ATIS dataset with SNIPS dataset as the training set. In this setting, AirlineTravel in ATIS is considered as the target domain while all of the seven domains in SNIPS are taken as the source domains. Our model still outperforms the existing coarse-grained slot type prediction by a large margin, especially in zero-shot setting, which shows that our model can fully make use of the semantic information encoded by queries and has the ability to recognize unseen slots while the baseline models fail to predict unseen slots in the irrelevant target domain AirlineTravel.

### Analysis on Unseen Slots and Unseen Sentences

To further study the effectiveness of our approach on zero-shot setting, we also conduct analysis on unseen slots and unseen sentences in target domains of SNIPS. Following [Liu et al., 2020b], we separate the test set on each domain into “seen sentence” and “unseen sentence”. An utterance is categorized into the “unseen sentence” part as long as there is an unseen slot in it. Otherwise, it is categorized into the “seen sentence” part. However, [Liu et al., 2020b] can not show the real zero-shot scenarios directly because a sample with both seen slots and unseen slots would be categorized into “unseen sentence” part in their experiments. Therefore, we recalculate the F1-scores for each slot separately instead of each sentence. In our experiments, if a slot does not exist in the remaining six source domains, it will be categorized into the “seen slot” part. Otherwise we categorize it into the “seen slot” part.
Table 4: F1-scores (%) on SNIPS (SN) and ATIS (AT) for different target domains under zero-shot setting. “w/o T” denotes that we directly test on the pre-trained BERT without fine-tuning on it, and “w/o PT” represents that we train our model from scratch without pre-training.

Table 5: Results of different types of queries under zero-shot setting. Desc., Trans., and Exp. indicate queries based on slot description, back-translation, and examples, respectively.

5 Conclusion and Future Work

In this paper, we propose a novel MRC framework to address cross-domain slot filling. Our approach comes with two key advantages: (1) the well-designed queries encoding significant prior knowledge about slot names; (2) being capable of utilizing the semantic information of BERT pre-trained on the MRC dataset SQuAD2.0. Our method obtains new state-of-the-art results on SNIPS and ATIS datasets in the cross-domain setting, which demonstrates its effectiveness. In the future, we would like to explore how to jointly address intent detection and slot filling tasks using a unified MRC framework in cross-domain scenarios.

Acknowledgments

The research work described in this paper has been supported by the National Key R&D Program of China (2019YFB1405200) and the National Nature Science Foundation of China (No. 61976015, 61976016, 61876198 and 61370130). The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve this paper.
References


