Incorporating rules into end-to-end dialog systems

Evgeniia Razumovskaia
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213
erazumov@cs.cmu.edu

Maxine Eskenazi
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213
max+@cs.cmu.edu

Abstract

End-to-end dialog systems are usually brittle when dealing with unexpected input. Since rule-based dialog systems are unable to generate novel utterances in response to a given context, they produce dull interactions with the user. This paper describes a generative system that is less brittle when dealing with novel input by proposing two methods of incorporating rules into end-to-end neural systems. Results show significant improvement in per-turn accuracy over baselines. In addition, the models demonstrate faster convergence in training, improved performance in limited data scenarios and higher diversity of output. This approach can be easily incorporated into any model that uses a general encoder-predictor pipeline.

1 Introduction

Task-oriented dialog systems help users achieve a goal using natural language. The goal can vary from restaurant search (Henderson et al., 2014) and bus information (Raux et al., 2005) to booking plane tickets (Asri et al., 2017). Historically, dialog systems were built as a pipeline with separate modules for context representation, belief state and generation connected to one another (Wen et al., 2016). The connection and composition of the modules introduces noise and complexity, e.g., the definition of dialog state and how much dialog history to maintain is a complex design decision to be made by the researcher (Williams et al., 2017). In addition, when constructing dialog systems in a modular fashion, it is expensive and time-consuming to collect intermediate labels.

Recent advances in neural networks have made it possible to construct dialog systems directly from the raw text input, in an end-to-end fashion. This is now one of the most frequently used approaches in dialog system training, both for chit-chat and task-oriented systems (Eric and Manning, 2017; Zhao et al., 2017; Serban et al., 2017, 2016). An advantage of this method is that the intermediate modules do not require annotation. However, neural networks perform well only on dialogs which are similar to the ones in the training set. This makes the systems brittle and not usable in practice in commercial settings. Another disadvantage of end-to-end systems is that they tend to generate repetitive and generic utterances (Li et al., 2016a,b).

Commercially deployed systems rely on large sets of manually-created rules (Williams et al., 2017). These rules make the system output less brittle in the face of unexpected user input and provide developers with the opportunity to add new constraints to the system as they are needed going forward. However, the downside of using these rules is that the variability of the output is reduced since they are applied as a mask on top of predicted candidate probability.

Recent effort has concentrated on combining rule-based and end-to-end approaches (Williams et al., 2017; Liang and Yang, 2018), but as yet there has not been a significant improvement on real user dialogs (as opposed to simulated users). This paper proposes two methods of incorporating rules into end-to-end neural systems. First, the rules can be used as a way to inject expert knowledge or to impose constraints or commonsense restrictions on the system. Second, the rules can be used

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

as masks which represent the actions that can or cannot be taken following a given context. The constraints in this paper are commonsense, e.g., if no options are found in the database, the system should not try to present any to the user.

The key idea here is to incorporate rules in an end-to-end neural dialog system so that the system can extract a maximal amount of information. The rules are used either as additional input or output and they are incorporated into the training procedure. These approaches are evaluated on the DSTC2 dataset (Henderson et al., 2014) to determine whether they improve performance on real-world dialogs. Experimental results show that the resulting system performs significantly better than baselines. In addition, the models show faster convergence in training, improved performance in limited data scenarios and higher diversity of output. This demonstrates the effectiveness of the approaches proposed in this paper for incorporating useful information from rules.

2 Related work

End-to-end systems

End-to-end dialog systems directly map the dialog history to the sequence of output words. Usually, Recurrent Neural Networks (RNNs) are used for sequence encoding and prediction (Serban et al., 2016; Lowe et al., 2017; Xu et al., 2016; Serban et al., 2017; Yao et al., 2015; Sordoni et al., 2015; Lei et al., 2018). These models are often applied to task-oriented dialog systems, adding an `api_call` to the set of possible actions. The database output retrieved by the `api_call` can then be enumerated as a sequence (Bordes et al., 2016) or incorporated into next utterance prediction using memory-based networks (Madotto et al., 2018). The RNN then learns to predict the next utterance using either a decoder based on the hierarchical encoder of the context(Sordoni et al., 2015), memory networks (Sukhbaatar et al., 2015), query regression networks (Seo et al., 2016) or copy-augmented networks (Eric and Manning, 2017). The RNN then produces output either by generating the words one by one or by ranking the response pool.

External knowledge in dialog systems

Previous work on incorporating external knowledge into end-to-end dialog systems can be divided into two parts – one where external knowledge is raw text and a separate model is trained to extract useful features from it and another where external knowledge is a curated set of features or of information known to be useful for the model, e.g., the bus schedule for a system providing bus information (Raux et al., 2005).

Lowe et al. (2015) is an example of the first approach, where an unstructured text was used as the source of additional information to improve next utterance retrieval. The unstructured text was encoded either using an RNN encoder or in TF-IDF topic terms. Using the RNN to encode knowledge and then passing it into the context encoder shows significant improvement in the model's performance .

The latter approach was used in Young et al. (2018) where the end-to-end structure is complemented by the structured commonsense knowledge base, ConceptNet (Singh et al., 2002), which includes formal facts like "Paris is the capital of France", as well as informal ones like "A dog is a pet". This has been shown to be effective for open domain dialog systems. However, for a task-oriented system, commonsense knowledge included in knowledge bases such as ConceptNet is too general to be helpful.

More recently, Hybrid Code Networks (HCNs) were proposed (Williams et al., 2017). They are designed to incorporate the developer's narrow domain knowledge into the model's output by masking the action templates that are deemed by the developer to not be logical, e.g., by keeping the model from asking the user about their preferred cuisine when the cuisine is already known. This approach has been shown to be effective for simulated datasets. Incorporating rules was not effective on real user interactions.

Several extensions to HCNs have been proposed. Liang and Yang (2018) proposed Hierarchical HCNs where a hierarchical recurrent network was used to encode each user utterance by first encoding each word at the character level and then passing the word encodings into the RNN to obtain the utterance embedding. Ham et al. (2019) extended HCNs with several trainable units to decrease the need for

expert knowledge in the dataset. Namely, they trained the entity-tracking and entity-filling units from the data. Liang and Yang (2018) and Ham et al. (2019) showed improved results on user-generated and simulated datasets, respectively. Lee (2018) modified the rules that are used to guide the choice of actions. A given action was masked based on the action predicted at the previous step and the action at the following step. These methods will need to be tested in the context of real user-machine interaction.

To summarize, previous work on incorporating external commonsense knowledge has not yet been shown to be effective for task-oriented systems in real conversations. Previous systems used rules only on system output. To our knowledge, the work described in this paper is the first attempt to create a hybrid system that uses rules in training to help predict the next system utterance.

3 Methods

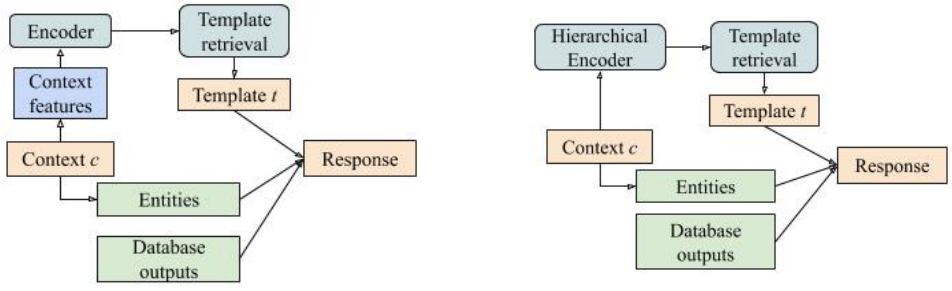
This section describes the methods which will be used for dialog response prediction. In addition to baseline methods for response prediction and generation, several methods for rule incorporation are described. It also describes the rule extraction algorithm.

3.1 Baseline models

Two baseline algorithms for response retrieval are explored – Hybrid Code Networks (HCNs, Williams et al. (2017)) and Hierarchical Recurrent Encoder Retrieval model. The latter turns the Hierarchical Recurrent Encoder Decoder (Serban et al., 2017) into a retrieval model. The two baselines are depicted on Figure 1. The baseline models do not incorporate rules neither in training nor in prediction.

The models differ in the method of encoding the dialog context, c. In *Hybrid Code Networks* (Fig. 1a) the context in encoded into 837 dimensional vector. The first 13 dimensions are high level hand-crafted features, e.g., whether a slot has been filled within the context or last user utterance. The full list of features is presented in the appendix A. The other 824 are a concatenation of two encodings of the last utterance – as a bag-of-words and as the average of word2vec vectors of words in the utterance. The context, over time, is encoded by passing the extracted features through a Long Short Term Memory network (LSTM) (Hochreiter and Schmidhuber, 1997). In contrast, *Hierarchical Recurrent Encoder Retrieval model* (fig. 1b) encodes the context using two LSTMs. As in Serban et al. (2017), one LSTM encodes each utterance into a vector while the other one encodes the context by processing the utterance vectors.

Response prediction is carried out in a template-based fashion. First a template is chosen out of the template pool. Then the slots in the template are filled with the values extracted from the context and obtained from the database. The filled template is the system response. The algorithm for template extraction is described below in section 4.1.



(a) Hybrid Code Network (Williams et al., 2017)

(b) Hierarchical Recurrent Encoder Retrieval model

Figure 1: A diagram of the baseline retrieval algorithms.

3.2 Rules

We incorporate the rules into the end-to-end dialog system training. Here, we describe the rules which will be used in training together with the dialog context.

The rules are constructed based on context features. Each rule is a binary vector masking out the templates which, according to the rules, should not be output. The rules were constructed based on

Stage of the dialog	What to do
No slot has been filled (the beginning of the dialog)	• Any response can be output
If a slot is filled	• Not to greet the user again • Not to ask about the filled slot again
Database has been queried	• Not to greet the user • Not to ask about any of the slots • Not to do an api call to the database
No results were obtained from the database	• Not to greet the user • Not to ask about any of the slots • Not to do an api call to the database • Not to attempt to return any information about the restaurants
Any results obtained from the database are available for presentation to the user	• Limit the outputs to those where an entry is presented

Table 1: The rules used, where the output is limited depending on the stage of the dialog

common sense and expert knowledge of the structure of the dialogs, e.g., that a slot will not be asked for again once it was filled or that the system should not greet the user once any of the slots have been filled. The rules are presented in more detail on Table 1. In practice, the rules are realized as the masks of templates presenting the response we would like the algorithm to output or avoid using.

3.3 Hybrid models

The aim is to combine the above rules with the end-to-end systems used as the baselines. This section shows how the rules are encoded to be used in training and how the rule encoding and baseline systems are combined.

3.3.1 Rule Encoder and Predictor

Rule encoding and predicting is performed using a feedforward layer with ReLU activation (Nair and Hinton, 2010). In encoding, the encoder layer maps the rules into the dimensionality of the output of the context encoder so that the error between them can be easily calculated (Section 3.3.2 provides more detail on this) .

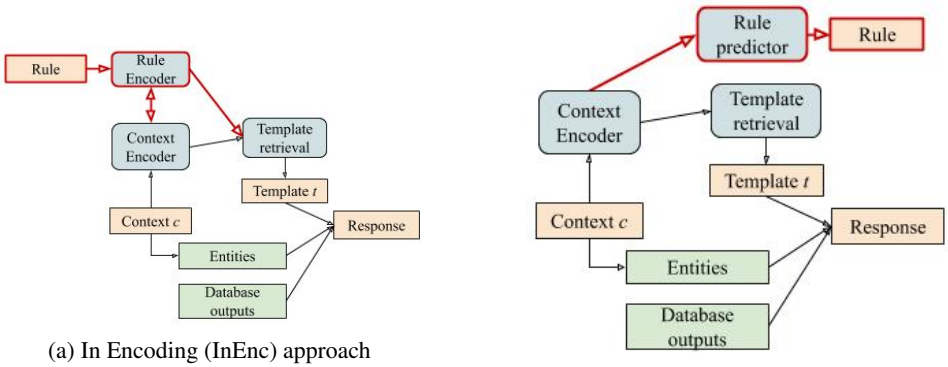
For the rule predictor, the task is multi-label multiclass classification since it aims to predict all templates that can follow a given context.

3.3.2 Combining rules and context encoders

The hybrid networks shown on Fig. 2 combine the rules with the end-to-end systems for the task of dialog response generation. These hybrid networks extract additional information from the rules that were pre-defined by the expert. In this way, experts can add knowledge to the end-to-end system and influence its performance and behavior. Main benefit of incorporating the rules into training (rather than applying them as the masks on top of the output) is that it does not only limit possible outputs but also forces the model to redistribute the probability across possible outputs.

We present two approaches for combining the end-to-end system with the rules – one where the rules are used **in encoding (InEnc)** and one where the rules are used **in prediction (InPred)**. The algorithms are presented on Fig. 2a and Fig. 2b, respectively. The new components introduced in the hybrid systems are shown in red. Next, the two approaches are described in detail.

Our **InEnc** approach incorporates rules while encoding the context. The intuition here is: assuming the rules provide good guidance for the pool of possible next utterances, the context encoder trained



(a) In Encoding (InEnc) approach

(b) In Prediction (InPred) approach

Figure 2: The rule incorporation algorithms. The two different context encoders presented on Fig. 1 are combined into a context encoder module in this Figure.

with the incorporation of the rules should be able to predict the correct utterance and also allocate probabilities to other options described by the rule.

The rules are incorporated by learning coordinated representations of the context and rules, i.e., by making the representations of the rules and contexts be as similar as possible. In other words, the aim is to maximize the cosine similarity between the encoding of the rule and the context. The loss used to maximize the cosine similarity is as follows: $loss(x, y) = 1 - cos(x, y)$ where $cos(x, y)$ is the cosine similarity between two input vectors x and y. Rule encoding is also passed into the template retrieval module by concatenating context encoding with rule encoding. Thus, in this case, a rule is incorporated into the model two times – when training the context encoder and when passing the rules to template retrieval. We use the rules twice because in the former approach the rules influence context encoding and not the prediction process directly, but in the latter approach they influence prediction and not context encoding. Thus, the combination incorporates the rules in both modules.

The **InPred** approach incorporates the rules in multitasking prediction, i.e., the rules are predicted simultaneously with next utterance prediction. In this approach, the system has to predict both the rule mask and the correct response from the same context encoding. Intuitively this means that context encoding is trained in such a way that it contains both the information about the concrete correct response and about the other possible candidates. This is achieved by backpropagating two losses through the system – one controlling the correctness of the predicted rule and the other one controlling the correctness of the predicted template. The shared part through which both of them are backpropagated is the context encoder, enforcing the intuition described above. The loss for the correctness of the predicted rule is binary cross-entropy. The loss for the correctness of the template is categorical cross entropy: $loss(y, \hat{y}) = -\sum_j \sum_i y_{ij} * log(\hat{y_{ij}})$ where y is the correct label and $\hat{y}$ is the predicted label. In this approach, the rules only influence the parameters of the context encoder but not the template predictor.

4 Experiments

4.1 Dataset

The task-oriented DSTC2 dataset (Henderson et al., 2014) was used. It contains human-machine dialogs collected using Amazon Mechanical Turk that are for the restaurant domain. A unique characteristic of this dataset is that the human-machine interaction only uses 2407 possible responses. The dataset statistics are given on Table 2. The complexity of these human-machine dialogs lies in the fact that they are noisier than simulated ones and involve ASR errors or misinterpretations, which could lead to unexpected system replies.

The refined version of DSTC2 was used Bordes et al. (2016). This version ignores the dialog state annotations and only includes the raw text. Raw text includes the user's and the system's utterances as well as the API calls and database outputs. The users could change their goal in the course of the dialog. Thus, once the user request was updated, the system could query the database again.

The templates are obtained by substituting the slot values in the 2407 candidates with substitute tokens. For instance, both "Golden_Wok is a nice restaurant in the north of town serving tasty Chinese

Feature	Value
Training dialogs	1618
Validation dialogs	500
Test dialogs	1118
Average num. of turns	9.22
Total number of slots	8
Number of informable slots	3
Number of requestable slots	5

Table 2: Statistics of the DSTC 2 dataset (Henderson et al., 2014)

food" and "Anatolia is a nice place in the centre of town serving tasty Turkish food" would be reduced to "`R_name` is a nice place in the `R_location` of town serving tasty `R_cuisine` food".

One feature of the dataset is that the training, development and test sets were collected using different dialog policies. This makes this dataset complex for machine learning algorithms since different replies are generated from identical contexts in the training, development and test sets. In this paper, the initial training, development and test splits given in Henderson et al. (2014) were followed, incorporating the full set of action templates.

Two dataset-specific metrics were used to evaluate the models. Template accuracy was used to measure how well the system predicts templates, i.e., making sure it does not fail at the first stage. Second, we measure the per-turn accuracy, a metric often used with this dataset (Eric and Manning, 2017; Madotto et al., 2018; Liu and Lane, 2017). This metric measures exact alignment of the predicted utterance to the reference response. This gives us an idea of the slot-filling correctness as well as the correctness of the predicted template.

4.2 Experimental setup

The hyperparameters used in the experiments where the HCNs were the base match the ones in Williams et al. (2017): hidden size of 128, tanh activation function and LSTM as the recurrent network (Hochreiter and Schmidhuber, 1997). In the experiments with HRE, the hyperparameters were the following: embedding size of 300, hidden layer size of 128, LSTM as the recurrent network and tanh as the activation function. All models were trained for 20 epochs using the Adam optimizer (Kingma and Ba, 2014) with 0.01 learning rate and a batch size of 8. In the experiments, the 11 rules listed on Table 1 were used.

4.3 Results

Table 3 shows results on the development and test sets. All improvements achieved by incorporating rules are statistically significant with 99% confidence. Compared to the HCN baseline, HRE with InPred rule incorporation obtains the largest 2.65% per-response accuracy improvement with a similar improvement in the template prediction accuracy.

We observe more improvement when the rules are applied to HRE than when applied to HCNs. The reason for this is that the main benefit of using rules (as explained in section 3.3.2) is their influence on the context encoder parameters. In HCN encodings, the context is first featurized in a predefined non-parametric way and then the input features are passed through the recurrent network. In HRE encodings, the context encoder is fully parametric. Thus, the rules have larger room for influence.

	Dev accuracy		Test Accuracy	
	T	PR	T	PR
HCN	54.62	45.66	54.87	46.99
HCN + InEnc	54.46	45.41	55.42	**47.91**
HCN + InPred	54.70	45.56	**55.83**	47.43
HRE	67.83	55.43	55.21	46.10
HRE + InEnc	68.50	55.81	**57.48**	**49.64**
HRE + InPred	69.38	57.10	57.16	48.27

Table 3: Evaluation results on the development and test sets. The values in bold indicate best performance. T is template accuracy. PR is the per-response accuracy (filled in templates).

We observe that that the largest improvement in template accuracy with the HCN encoder was obtained with +InPred while the largest improvement with the HRE encoder was obtained with

+InEnc. Although it is impossible to pin down the cause of this difference, we assume that it lies in the difference between the context encoders. When using +InPred, the model is forced to extract as much information as possible from the given input features. When using +InEnc, the main effect is in the context representations since context and rule representations are encouraged to be similar. Thus, when context representations are obtained from the raw data in an automated way (using an ML model), the +InEnc method has a stronger influence over the context encoder. In contrast, when the context representations are a set of predefined features, +InPred has more space for improvement since it allows the model to use input features in an optimal way.

These significant gains in per-response and template accuracy confirm our hypothesis that the incorporation of rules in training end-to-end dialog systems leads to improved modelling of the dialog.

5 Discussion

This section presents the analysis of the proposed methods from three perspectives – convergence, limited training data and diversity of the output.

5.1 Convergence

The hypothesis here is that the system with rules obtains additional guidance in determining which action to take at each stage of the dialog. Thus additional guidance should make it easier for the system to learn from the data.

Experimental results are presented on Fig. 3. This figure shows per-response accuracy across the training epochs. With Hybrid Code Networks, the model that does not use rules obtains best results at epoch 8 and then fluctuates. The model with InPred converges after the 4th epoch. The model with InEnc does not converge earlier than the two methods described above – it converges at epoch 10 but obtains better performance. With the Hierarchical Recurrent Encoder, the model without rules converges after the 6th epoch. The model with InEnc converges after the 2nd epoch. Thus, earlier convergence is observed when rules are incorporated both for parametric and non-parametric context encoders.

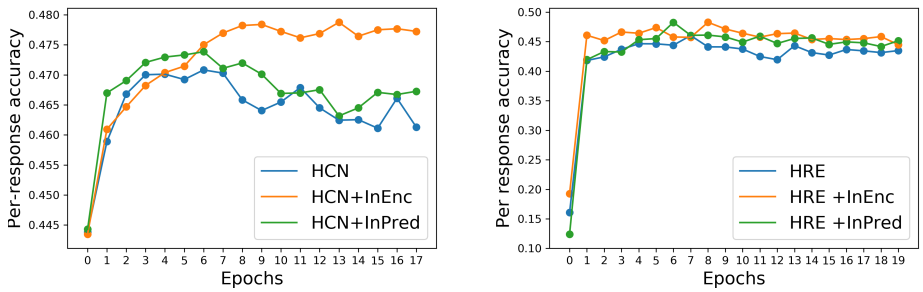


Figure 3: Performance on the test set across training epochs for HCNs and HRE.

5.2 Limited training data

The hypothesis here is that the algorithms incorporating rules will outperform the baseline systems due to the explicit guidance as to which utterance should be output next. We assume that the baseline systems would require a lot of data to learn the dependencies between the context and possible output templates while the ones with the rules obtain this information explicitly.

The results presented on Fig. 4 show per-response accuracy as a function of the percentage of training data used. We see that models with rules improve better with the HCN baseline and the difference is not significant for the HRE based models. There is an order more parameters to be learned in HRE than in HCN. Thus in this case it is difficult for the model to learn even simple patterns from a small training set. To control the variance of small datasets, the same experiment with 5 different splits was run. The results are presented in app. B. The HCNs results suggest that the use of rules lessens the amount of training data required.

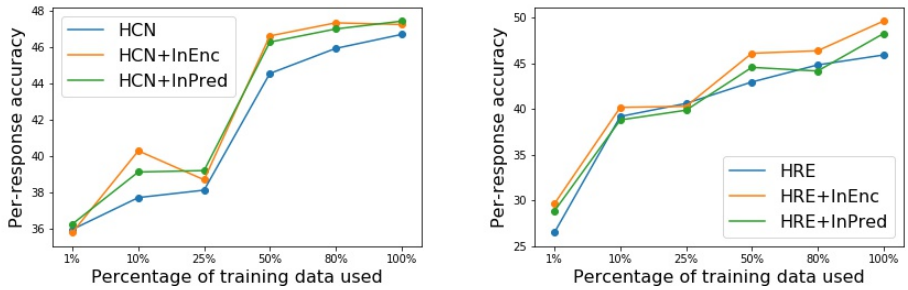


Figure 4: Variation of per-response accuracy as a function of different percentages of training data

5.3 Diversity of the output

One of the aims of creating systems that incorporate rules is to produce more varied predicted outputs. Here we present the qualitative analysis of the diversity of the output.

Intuitively, when the rules are incorporated into training, increased variety of responses is obtained since the rules force the model to redistribute the probabilities to assign higher probabilities to the templates which are in the pool of possible responses defined by the rule.

We studied the cases where the HRE +InEnc system performed better than the HRE system. Systematic improvement was observed for utterances requesting additional information from the user. As shown in the examples in Table 4, the two policies use very different utterances to request additional information. They could not be reduced to the same template. The improvement in this case comes from the incorporation of rules because otherwise the model has no incentive to predict an unseen class.

User	System
...	
I'm looking for a cheap restaurant that serves Italian food	***What part of town do you have in mind?***
...	

User	System
...	
any type	***There are restaurants if you don't care about the food . What area do you want?***
...	

Table 4: Excerpts from dialogs in training set (on the left) and test set (on the right)

6 Conclusions and future work

This paper presents two approaches for incorporating rules (expert knowledge) in the training of end-to-end dialog systems. We propose the InEnc method which incorporates the rules into the encoding of the dialog context and obtains strong performance improvement on the DSTC2 dataset. The methods are demonstrated to converge faster than baselines, to be effective in the limited data scenarios and to output more diverse utterances than the baseline.

The appeal of the methods lies in the fact that any end-to-end dialog system can be complemented with these rule-incorporation methods. This is due to the fact that they can be applied to any system that uses an encoder-predictor pipeline, which is the case for most recent models. This aspect will be examined in future work. We are also planning to develop supervised and unsupervised approaches for obtaining the rules in an automated fashion since it is difficult or infeasible to create rules manually when the candidate set is large, as is the case for many of the human-to-human or Wizard of Oz datasets.

Acknowledgements

The authors would like to thank Dr. Tiancheng Zhao for deep and interesting discussions and advice on the project.

This paper has been supported by a grant from the AI Center, Samsung Research. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Samsung Research.

References

Asri, L. E., Schulz, H., Sharma, S., Zumer, J., Harris, J., Fine, E., Mehrotra, R., and Suleman, K. (2017). Frames: A corpus for adding memory to goal-oriented dialogue systems. *arXiv preprint arXiv:1704.00057*.

Bordes, A., Boureau, Y.-L., and Weston, J. (2016). Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*.

Eric, M. and Manning, C. D. (2017). A copy-augmented sequence-to-sequence architecture gives good performance on task-oriented dialogue. *arXiv preprint arXiv:1701.04024*.

Ham, J., Lim, S., Lee, K.-H., and Kim, K.-E. (2019). Extensions to hybrid code networks for fair dialog dataset. *Computer Speech & Language*, 53:80–91.

Henderson, M., Thomson, B., and Williams, J. D. (2014). The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 263–272.

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

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lee, S. (2018). Nudging neural conversational model with domain knowledge. *arXiv preprint arXiv:1811.06630*.

Lei, W., Jin, X., Kan, M.-Y., Ren, Z., He, X., and Yin, D. (2018). Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1437–1447.

Li, J., Galley, M., Brockett, C., Gao, J., and Dolan, B. (2016a). A diversity-promoting objective function for neural conversation models. In *Proceedings of NAACL-HLT*, pages 110–119.

Li, J., Monroe, W., Ritter, A., Jurafsky, D., Galley, M., and Gao, J. (2016b). Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202.

Liang, W. and Yang, M. (2018). Hierarchical hybrid code networks for task-oriented dialogue. In *International Conference on Intelligent Computing*, pages 194–204. Springer.

Liu, B. and Lane, I. (2017). An end-to-end trainable neural network model with belief tracking for task-oriented dialog. *Proc. Interspeech 2017*, pages 2506–2510.

Lowe, R., Pow, N., Serban, I., Charlin, L., and Pineau, J. (2015). Incorporating unstructured textual knowledge sources into neural dialogue systems. In *Neural Information Processing Systems Workshop on Machine Learning for Spoken Language Understanding*.

Lowe, R. T., Pow, N., Serban, I. V., Charlin, L., Liu, C.-W., and Pineau, J. (2017). Training end-to-end dialogue systems with the ubuntu dialogue corpus. *Dialogue & Discourse*, 8(1):31–65.

Madotto, A., Wu, C.-S., and Fung, P. (2018). Mem2seq: Effectively incorporating knowledge bases into end-to-end task-oriented dialog systems. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1468–1478.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

Raux, A., Langner, B., Bohus, D., Black, A. W., and Eskenazi, M. (2005). Let's go public! taking a spoken dialog system to the real world. In *Ninth European conference on speech communication and technology*.

Seo, M., Hajishirzi, H., and Farhadi, A. (2016). Query-regression networks for machine comprehension. *arXiv preprint arXiv:1606.04582*.

Serban, I. V., Sordoni, A., Bengio, Y., Courville, A., and Pineau, J. (2016). Building end-to-end dialogue systems using generative hierarchical neural network models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Serban, I. V., Sordoni, A., Lowe, R., Charlin, L., Pineau, J., Courville, A., and Bengio, Y. (2017). A hierarchical latent variable encoder-decoder model for generating dialogues. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Singh, P., Lin, T., Mueller, E. T., Lim, G., Perkins, T., and Zhu, W. L. (2002). Open mind common sense: Knowledge acquisition from the general public. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 1223–1237. Springer.

Sordoni, A., Galley, M., Auli, M., Brockett, C., Ji, Y., Mitchell, M., Nie, J.-Y., Gao, J., and Dolan, B. (2015). A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 196–205.

Sukhbaatar, S., Weston, J., Fergus, R., et al. (2015). End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Wen, T.-H., Vandyke, D., Mrksic, N., Gasic, M., Rojas-Barahona, L. M., Su, P.-H., Ultes, S., and Young, S. (2016). A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*.

Williams, J. D., Asadi, K., and Zweig, G. (2017). Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 665–677.

Xu, Z., Liu, B., Wang, B., Sun, C., and Wang, X. (2016). Incorporating loose-structured knowledge into lstm with recall gate for conversation modeling. *arXiv preprint arXiv:1605.05110*, 3.

Yao, K., Zweig, G., and Peng, B. (2015). Attention with intention for a neural network conversation model. *arXiv preprint arXiv:1510.08565*.

Young, T., Cambria, E., Chaturvedi, I., Zhou, H., Biswas, S., and Huang, M. (2018). Augmenting end-to-end dialogue systems with commonsense knowledge. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Zhao, T., Lu, A., Lee, K., and Eskenazi, M. (2017). Generative encoder-decoder models for task-oriented spoken dialog systems with chatting capability. *arXiv preprint arXiv:1706.08476*.

Appendices

A Context features

Feature	Number
Presence of each entity in the dialog context	3
Presence of each entity in the the current utterance	3
Has the DB been queried?	1
Are the DB results empty?	1
Are the DB results non empty?	1
Have the DB results been presented?	1
Have all the DB results been presented?	1
Are there any DB results which have not been presented yet?	1
Does the current api_call retrieve no results in the training set?	1
Does the cuisine in the current utterance retrieve no DB results in the training set?	1

Table 5: The features extracted from the dialog context dialog. From Williams et al. (2017)

B Average results across 5 different subsamples of the data for HCN encoder

Data proportion	Baseline	+InEnc	+InPred
2%	36.928832 $\pm$ 0.197006	37.089160 $\pm$ 0.146753	36.962679 $\pm$ 0.189317
10%	40.400819 $\pm$ 0.095001	40.578961 $\pm$ 0.148661	40.336688 $\pm$ 0.234349
25%	40.431103 $\pm$ 0.135481	40.128262 $\pm$ 0.477918	40.475639 $\pm$ 0.106647
50%	45.860871 $\pm$ 0.194689	46.898102 $\pm$ 0.241756	46.365903 $\pm$ 0.241109
80%	46.662510 $\pm$ 0.138493	46.865592 $\pm$ 0.164103	47.081143 $\pm$ 0.167623
100%	47.189810 $\pm$ 0.195307	47.860132 $\pm$ 0.096755	47.442771 $\pm$ 0.071833

Table 6: Means and standard deviations of 5 runs with different splits of the reduced data