Structure in Dialog

Shrimai Prabhumoye
sprabhum@andrew.cmu.edu

Samridhi Shree Choudhary
sschoudh@cs.cmu.edu

1 Introduction

This document presents the design and implementation of a sequence-to-sequence model used to understand the structure of 'Open-Domain Dialogs'. There has been a recent surge in research related to conversational agents and dialog systems ((Li et al., 2016), (Vinyals and Le, 2015), (Vlad Serban et al., 2016)). With efforts ranging from rule-based dialog systems to deep neural networks for both task-oriented and open-domain dialog agents, the primary focus has been on making dialog with these agents more relevant, coherent, informative and interesting. However, there is a significant gap in generalizing the generation of responses to a user query, especially in an open domain environment (Li et al., 2016). A majority of contemporary systems handle this by responding with a maximum likelihood response present in their training data. Some examples of these monotonous responses are: '*I don't know*', '*I think so*' and so on depending on the training data (Li et al., 2015).

Users can ask different types of questions. There is a need to understand the general structure of responses to these queries. This structural information can lead to better coherence in the responses generated. Our work is an attempt in this direction. We take the first step towards understanding the inherent structure of the responses. The work presented in this document focuses on understanding the underlying structure of the response and partially generating the response. We plan to focus on generation of the entire response in the future work. Following sections detail the goal of our current effort and a description of the data used. We give an overview of our methodology employed and present the experiments and the results performed by us. We conclude with discussion on the results and our plan to extend this in future.

2 Motivation

In order to understand the structure of the response, we attempt to extract and learn the 'templates' present in the responses in our training data. Templates aim to segment the responses into a functional and a content section. The functional sections serve as an introductory stage to the rest of the response. In other words they represent the inherent style of the responses. Hence, a '**template**' denotes the style in which the content of the response is introduced.

We attempt to extract templates from the responses and learn these templates for the given query-response pair. We use an encoder-decoder model with and without attention. We expect the model to be able to learn the style of response for different types of questions asked and ultimately use this information to generate the response style for a question not seen in the training data. Following are the main goals or the research questions we plan to work on for this project:

- Understand the structure of the dialog responses for Open Domain dialogs.
- Segment the response sentences into '*Content*' and '*Introduction Style*' to this content.
- Learn this structure and predict the response style for a given question.

3 Data

We used the Ubuntu Corpus (Lowe et al., 2015) for our experiments. The dataset consists of the speaker information, the dialog turn number, the words of the utterance and the dialog start and end markers.We constructed our dataset, from this corpus, that consists of a total of 963593 utterance-response pairs. The train, validation and test split

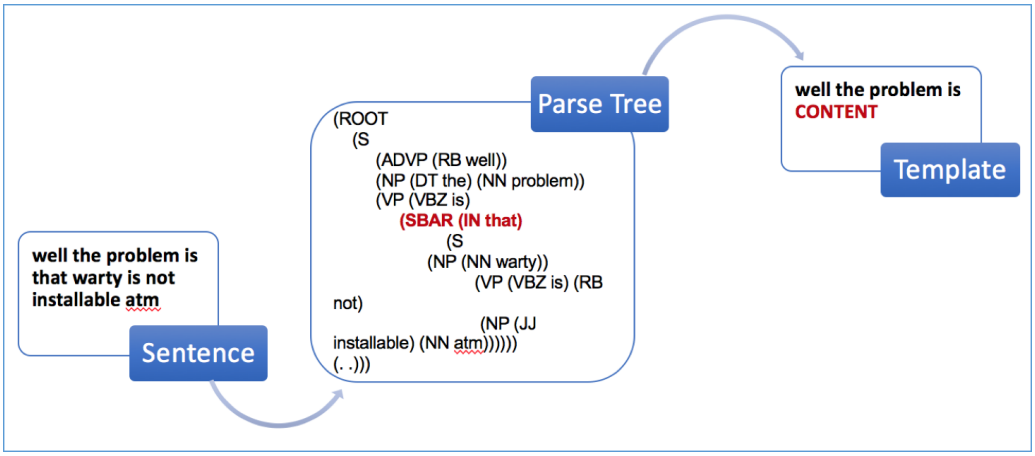

Figure 1: An Example Template

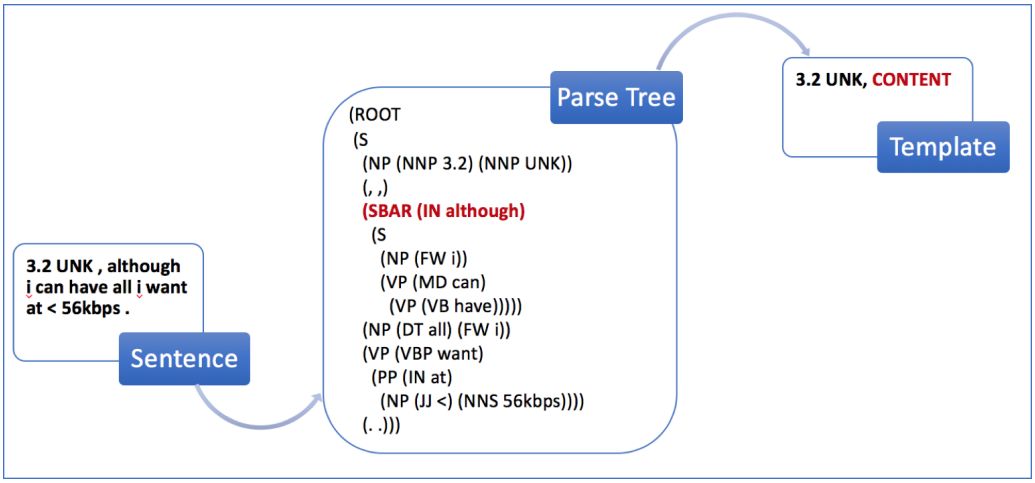

Figure 2: Noisy Template

used is 70%, 10% and 20% respectively. We assume that the query and response pairs of 2 speakers in the corpus contains alternating turns of query and responses. Hence, we constructed our dataset by taking the first turn of a dialog as the query and the next turn as the response to this query and so on.Only the first utterance of the response is considered for our experiments in order to observe the style in which the content of the response was introduced. This dataset is referred as 'Full' in our experiments.

4 Proposed Methodology

Following sections detail the method employed by us to understand the structure of the responses in our dataset.

4.1 Template Extraction

In order the understand the inherent style of the responses we divide the responses in two sections: 'style' and 'content'. The template for the response is created by concatenating the style and the content as: '***style* CONT**'. In the template, the content part of the response is replaced by the tag 'CONT'. In order to segment the functional section from the content of the response, we look at the parse trees generated for the responses. We use the Stanford Parser (Nivre et al., 2016) to create the parse trees. The parse labels in the parse tree capture the structure of the sentence. Among these labels, we specifically look for the presence of a '*clause*' in the sentence. This is primarily because, by definition, clauses are grammatical constructs representing a complete thought. The corresponding label in the parse tree for such a clause is '**SBAR**'. We follow the following steps in order to construct the template for the sentence:

- Parse the response and derive its parse tree.
- Search for the presence of the tag 'SBAR' to

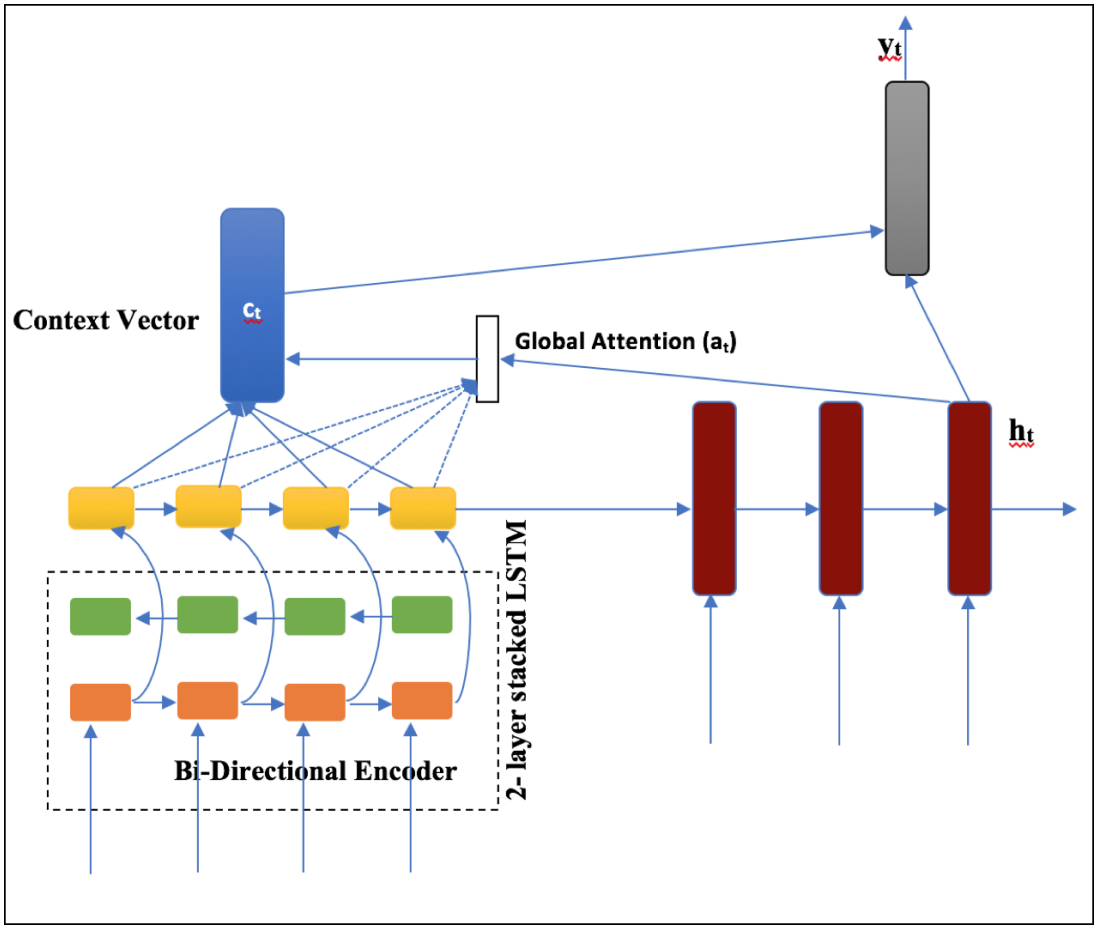

Figure 3: Encoder-Decoder Architecture with Global Attention

identify the clause present in the sentence.

- Segment the sentence at SBAR of the parse tree.
- Segment before *SBAR* = style
- Segment after *SBAR* = content
- Template = "***style* CONT**"

Figure 1 shows an example template for one of the sentences in our training set.

Due to the nature of our dataset, the above segmentation technique leads to some noisy templates too. Figure 2 shows an example of one such noisy template. These can confuse the model when it attempts to learn the styles for the corresponding questions since there is no coherence with the style extracted for the respond and the question at hand. Hence, we filter our dataset to get good templates.

Dataset. A majority of the responses did not have any template i.e. it did not have an introductory style for the content. Eg: 'sudo apt-get install X' will not have a template due to the absense of SBAR. After filtering the dataset, we get 48k query-template pairs. We create train, validation and test splits as 80%, 10% and 10% respectively from these 48k pairs. This dataset is referred to as 'Template' in our experiments.

4.2 Model Architecture

In order to learn the style of the responses for the questions asked, we use an encoder-decoder model for the parallel corpus created by us as described in Section 3. We experiment with different configurations of the dataset and with applying/removing global attention to the model employed (Bahdanau et al., 2014).

Figure 3 shows the architecture of the model used. The hyper-parameters for our setup are detailed in Table 1

We used a bi-directional LSTM with two layers for encoding the questions in our dataset.

Metric. We check the convergence of the model by calculating the perplexity and the accuracy on the validation dataset. The models are saved after

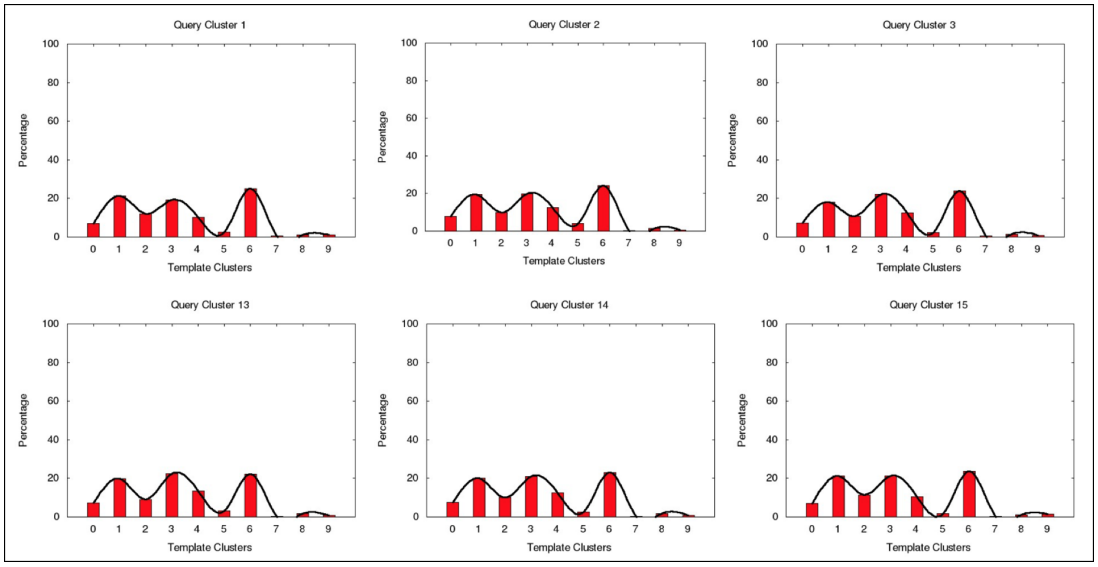

Figure 4: Templates Across Queries

every epoch. Accuracy is calculated as percentage of words that are correctly predicted by the model when compared to the words in the actual response. The perplexity is calculated as shown in the equation (1) below:

$$perplexity = e^{-\frac{1}{N}\sum_{i=1}^{i=N} \log p(x_i)} \quad (1)$$

The generations are performed with the best performing model separately after training has completed. We trained our models on GPUs and used mini-batching to speed up the training process.

Hyper Parameter	**Value**
Bi-Directionality	True
Encoder Hidden Size	500
Decoder Hidden Size	500
Embedding Size	300
Number of Layers for Encoder	2
Mini-Batch Size	32
Learning Rate	[0.01-0.05]
Training Epochs	40
Beam Size for generation	3
RNN type	LSTM
Trainer Type	SGD Trainer
Loss used for Training	Negative Log-Likelihood

Table 1: Hyper Parameters

4.3 Clustering of Queries and Templates

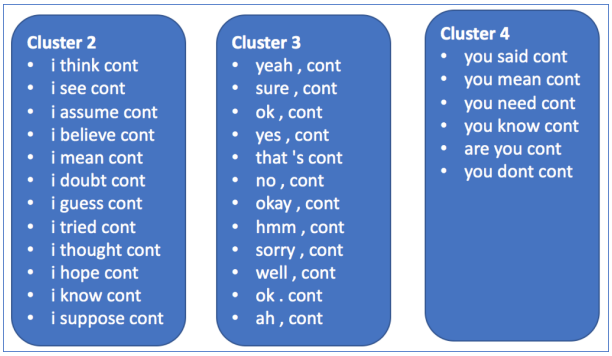

Figure 5: Query Clusters

In order to help us understand the distribution of templates across the different query types, we clustered the queries and the templates. We used 'Glove Embeddings' (Pennington et al., 2014) for each word in the vocabulary. A representation of the query and the template is calculated by taking an average of all the word-embeddings in a sentence.

Let query q_i have n words $(q_{i1}, q_{i2}, ..., q_{in})$ and the corresponding template t_i have m words $t_{i1}, t_{i2}, ..., t_{im})$. Then the representation of q_i and t_i is calculated as:

$$embed(q_i) = \frac{1}{n}\sum_{j=1}^{n} embed(q_{ij}) \quad (2)$$

$$embed(t_i) = \frac{1}{m}\sum_{j=1}^{m} embed(t_{ij}) \quad (3)$$

The dimension of the glove embedding used for each word is 300. Hence, the final representation of the query and the template is also of dimension 300. Such representations are calculated for all the queries and templates in the training set. These queries and templates are clustered using the representation. K-means was used to perform this clustering. We experimented with different number of clusters and created 20 query-clusters and 10 template-clusters.

The clusters of templates are shown in Figure 5. As we can see, we get well-formed coherent clusters. Next, we plot the distribution of template clusters for each of the query clusters. Some of the distributions of template clusters are shown in Figure 4

5 Experiments and Results

The multiple experiments performed with the setup described so far can be broadly categorized into three categories. Each of them is detailed in the following subsections. The results for each experiment is presented in the corresponding subsections. Further discussion on the results and the inferences that are drawn are presented later.

5.1 Seq-Seq on the Full Corpus

The full corpus consists of the raw question-answer pairs without any templates extracted. We estimated encoder-decoder models, with and without attention, on this dataset. We consider this as our primary baseline. The train and validation perplexity are reported in Table 2. Training was done using SGD learner and the results for the same can be seen in the table. A detailed discussion on the numbers are in Section 6.

5.2 Seq-Seq on Query-Template Corpus

As mentioned before, the full corpus was modified in order to identify and extract templates for the responses. The extraction was operationalized as detailed in Section 4.1. The parallel corpus of question-template pairs was used to estimate encoder-decoder models, similar to the baseline mentioned above, with and without attention. The primary motive of this model is to train it to predict the style or the template of the response, given any question from the test set. Table 2 shows the train and validation perplexity for these models.

5.3 Cluster Classification Model

For the template dataset, we calculated the representation of each query and template as described in Section 4.3. In this experiment, we tried to predict the cluster of the template using the encoded representation of the query. This is a classification task.

Let a query q_i have n words $w_{i1}, ..., w_{it}, ..., w_{in}$. Then the encoder LSTM equation for the time step t would be:

$$h_t = LSTM(embed(w_t), h_{t-1}) \quad (4)$$

We use pre-trained Glove embedding (Pennington et al., 2014) for the embedding of the words. Then for classification we follow:

$$\tilde{h_t} = h_t W_h + b_h \quad (5)$$
$$s_t = softmax(\tilde{h_t}) \quad (6)$$

Here $W_h \in R^{hidden\_size * numberofclusters}$, $b_h \in R^{1 * numberofclusters}$ and max() over s_t gives the predicted cluster of the template for the query q_i.

The results for the cluster classification task are shown in Table 3.

Since the results of the classification did not give much insight in the connection between the template clusters and the queries, we re-weighted the negative log likelihood loss. We first calculate the probability of each template cluster and the inverse of the probability is the weight of the cluster. Hence, the penalty for wrongly predicting a rare template cluster is high. The results for the cluster classification task are shown in Table 3

6 Discussions

As mentioned before, the model defined in Section 5.1 is considered as our baseline model. The model with attention performs slightly better than the one without attention, as visible in Table 2, in terms of both Validation and Accuracy.

Unfortunately, we face the maximum likelihood generation problem, initially mentioned by us, for the models for template generations. The templates generated by our models after finishing 40 epochs of training are the most frequently occuring templates in the training corpus: '**i CONT**' (that comprises of 8% of the training set templates) and '**i think CONT**' (that comprises 10% of the training set). In order to mitigate it to some

Model Type	Attention	Train Acc	Train Ppl	Validation Acc	Validation Ppl	Test Perplexity
Dataset: Full	Yes	28.32	47.67	27.25	55.14	81.05
Dataset: Full	No	28.20	49.06	27.12	56.26	-
Dataset: Template	Yes	69.21	3.24	69.26	3.24	5.01
Dataset: Template	No	68.87	3.39	68.84	3.39	-

Table 2: Experiment Results

Data Type	Reweight	Train Acc	Validation Acc
Dataset: Template	No	25.51	25.74
Dataset: Template	Yes	11.40	18.35

Table 3: Clustering Results

extent and help us understand the relationship between the questions and answers better, we performed clustering experiments, where the queries and the corresponding templates were separated into separate clusters.

The queries and templates were clustered as described in Section 4.3. The motivation behind this experiment was that the distribution of template clusters would be different for different query clusters. For example, let's say that we expect query cluster type 1 to have a high count of template cluster type 5 in the responses when compared to query cluster type 2. But what we see from the Figure 4 is that the distribution of template clusters is the same across all the query clusters. This leads us to believe that there is a weak connection between the queries and the templates. We further performed cluster classification experiments as described in Section 5.3. As shown in Table 3, re-weighting the classification loss in order to give higher penalty for wrongly predicting a rare cluster class did not improve the accuracy at all. We would like to explore further in this and revisit the re-weighting strategy.

Error Analysis. We performed manual analysis of the data after performing the clustering experiment. We noticed that some templates do not have much information in them. For example, 'i CONT' template can have different variations. It could be present as 'i think CONT' or 'i would say CONT'. A similar example is that of 'its CONT'. It is a noisy template. This brings us back to the heuristic we chose to create templates (segment at SBAR). Therefore, experimenting with different heuristics in order to get better templates makes sense.

Another reason for not finding connection between the query and templates is that the Corpus itself does not have the stylistic features that we are looking for. This is evident from the size reduction in our dataset from 960k to 48k when we filtered for the presence of valid templates. The remainder of the corpus did not have any templates or had extremely noisy templates as shown in Figure 2.

7 Conclusion and Future Work

As we see from the results, we get a 1.12 point improvement in validation perplexity by using attention model on the full dataset. We get a marginal 0.15 improvement in validation perplexity in the template dataset. The clustering experiment and the following classification of clusters experiment tells us that there is a poor connection between the queries and the templates.

It is an inherently difficult task to extract and learn structure in dialog. This is further aggravated for open domain dialog where the relationship between a query and a response becomes even more abstract. In other words the queries and the responses do not follow a set pattern and varies with the personality and the environment of the participants involved. We had hoped that in the Ubuntu Dialog Corpus, where the question-answers are fairly technical, the responses to the questions might have a structure imbibed in it that can be learned. However, as mentioned before, it was not very successful for the approach taken by us for this study.

We look to further investigate along different lines. We are hopeful that encoding structural information while training will help the model learn the dialog style better. In this direction we would like to experiment with including the CCG tags for

the words in the responses. CCG supertags gives the information about the kind of arguments expected by the word. Hence, they hint on the structure and the context of the tokens in the sentence. We would like to experiment with different styles of encoding the tag information during training and generation. We would also like to explore different heuristics for defining templates. Extracting generic templates would make the learning much more efficient and remains a challenging question to be answered. Apart from this, we would also like to explore for better datasets for open domain dialog.

8 Contributions

Both of us were involved in brainstorming the ideas for this project. We contributed equally in the implementation and debugging of the models. The individual contributions are detailed below.

Shrimai worked on:

- Template extraction
- Encoder-Decoder implementations
- Clustering
- Experiments and Error Analysis
- Report

Samridhi worked on:

- Template extraction
- Encoder-Decoder implementations
- Experiments and Error Analysis
- Report

References

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

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055* .

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

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *arXiv preprint arXiv:1506.08909* .

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*. pages 1659–1666.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869* .

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016. A hierarchical latent variable encoder-decoder model for generating dialogues. *arXiv preprint arXiv:1605.06069* .