Hierarchical Variational Memory Network for Dialogue Generation

Hongshen Chen†
Data Science Lab, JD.com
chenhongshen@jd.com

Zhaochun Ren†
Data Science Lab, JD.com
renzhaochun@jd.com

Jiliang Tang
Data Science and Engineering Lab
Michigan State University	
tangjili@msu.edu

Yihong Eric Zhao
JD.com
ericzhao@jd.com

Dawei Yin†
Data Science Lab, JD.com
yindawei@acm.org

ABSTRACT
Dialogue systems help various real applications interact with humans in an intelligent natural way. In dialogue systems, the task of dialogue generation aims to generate utterances given previous utterances as contexts. Among various spectrums of dialogue generation approaches, end-to-end neural generation models have received an increase of attention. These end-to-end neural generation models are capable of generating natural-sounding sentences with a unified neural encoder-decoder network structure. The end-to-end structure sequentially encodes each word in an input context and generates the response word-by-word deterministically during decoding. However, lack of variation and limited ability in capturing long-term dependencies between utterances still challenge existing approaches. In this paper, we propose a novel hierarchical variational memory network (HVMN), by adding the hierarchical structure and the variational memory network into a neural encoder-decoder network. By emulating human-to-human dialogues, our proposed method can capture both the high-level abstract variations and long-term memories during dialogue tracking, which enables the random access of relevant dialogue histories. Extensive experiments conducted on three large real-world datasets verify a significant improvement of our proposed model against state-of-the-art baselines for dialogue generation.

CCS CONCEPTS
• Information systems → Retrieval tasks and goals; Question answering;

KEYWORDS
Dialogue generation, Hierarchical Variational Memory Network, Recurrent Encoder-Decoder Model

1 INTRODUCTION
Dialogue generation aims to generate natural-sounding replies automatically to exchange information (e.g., knowledge, sentiments, etc.) and complete a variety of specific tasks in a conversation interaction process [35, 44]. In recent years, automatic dialogue generation has received increasing attention in numerous applications from e-commerce technical support to personal assistant tools [1, 2, 7, 29, 30, 36, 44]. Among all these approaches, end-to-end neural generation models [20, 21, 33, 34, 38, 41] have been proved to be capable in multiple dialogue system applications with promising performance. Most of these end-to-end neural generation models apply encoder-decoder architecture based on recurrent neural network, which directly maps an input context to the output response.

However, challenging problems still exist in current neural models for dialogue generation [6]: (1) Meaningless responses: Given a wide range of contexts, dialogue systems trained via neural generation models are still likely to generate short but pointless responses, such as “ha-ha” and “I don’t know”. Since these neural generation models are deterministic and shorter responses often have higher likelihood in posterior inference, these models inject no variation during the generation of reply [6]; (2) Memory decay: Vanishing memory is always a challenging problem in the task of dialogue generation. A dialogue session usually contains multiple turns, the longer the conversation moves forward, the harder to capture the long-term memorial dependencies. In particular, the injection of variability exacerbates the difficulty to capture the long-term memory. As a consequence, the response generation for long utterance context is extremely hard, which determines the response generation for long utterance context a rather difficult problem.

In this paper, we aim at developing methods that can tackle challenges of meaningless response and long-term memory dependencies, simultaneously. As an example, consider a dialogue session in Figure 1. Compared with Response 1, although both responses are suitable for the context, Response 2 is more meaningful than its counterpart. With the dialogue session moving forward, modeling the utterance dependencies becomes more and more difficult. As
Our main contributions can be summarized as follows:

- We identify the problems of response informativeness, and long-term utterance dependencies in dialogue generation.
- We build a hierarchical variational memory network model to generate reasonable, informative, and diversified responses by bridging the hierarchical architecture and variational memory network.

We build a hierarchical variational memory network model, i.e., HVMN, a hierarchical variational memory network (HVMN), which combines the spirits of variational autoencoder [18] and memory networks [39, 40] in a hierarchical recurrent neural network setting and aims to memorize the dialogue histories while retrieving the memory with a bit of randomness. The hierarchical structure encodes the utterance sequences, while the variational memory samples a latent variable for each utterance, conditioned on all the previous information, and then memory cells are retrieved by the latent variable. The memory cells keep updated to memorize the new utterance and the response generation decoder is guided by the variational memory block. We conduct extensive experiments on two benchmark datasets and an e-commerce custom-service dataset. Experimental results show that HVMN outperforms state-of-the-art baselines and generates more informative responses.

Our attempt to tackle above obstacles leads to a novel framework, i.e., HVMN, a hierarchical variational memory network (HVMN). It combines the spirits of variational autoencoder [18] and memory networks [39, 40] in a hierarchical recurrent neural network setting and aims to memorize the dialogue histories while retrieving the memory with a bit of randomness. The hierarchical structure encodes the utterance sequences, while the variational memory samples a latent variable for each utterance, conditioned on all the previous information, and then memory cells are retrieved by the latent variable. The memory cells keep updated to memorize the new utterance and the response generation decoder is guided by the variational memory block. We conduct extensive experiments on two benchmark datasets and an e-commerce custom-service dataset. Experimental results show that HVMN outperforms state-of-the-art baselines and generates more informative responses.

The proposed framework is validated by the extensive experiments and outperforms the state-of-the-arts on both metric-based and human evaluations.

The remaining of the paper is organized as follows. We formulate our research problem in §2 and describe our approach in §3. Then, §4 details our experimental setup and results. Related work is presented in §5. Finally, §6 concludes the paper.

2 PRELIMINARIES

In this section, we first formulate the dialogue generation task formally, and then introduce preliminaries of sequence to sequence models.

2.1 Task Definition

Before getting into the dialogue generation task, We introduce our key notations and concepts. Table 1 lists the main notation we use.

A dialogue session consists of a sequence of turns of utterances $D = U_1, ..., U_M$ between two interlocutors. At $m$-th turn, given previous utterances $U_1, ..., U_{m-1}$, the dialogue generation model aims to calculate the probability of $U_m$ given $U_1, ..., U_{m-1}$, i.e., $P(U_m | U_1, ..., U_{m-1})$.

Each utterance $U_m$ is a variable-length sequence of words, i.e., $w_{m,1}, ..., w_{m,n_m}$, where $w_{m,n}$ is the $n$th word of the utterance $U_m$, and $N_m$ is the length of utterance. $P(U_m | U_1, ..., U_{m-1}, w < n)$ is modeled by decomposing the probability distribution over both the previous utterances and the previously generated words:

$$P(w_{m,1}, ..., w_{m,n_m} | U_1, ..., U_{m-1}) = \prod_{n=1}^{N_m} P(w_{m,n} | U_1, ..., U_{m-1}, w < n).$$

Table 1: Glossary.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D$</td>
<td>a dialogue session</td>
</tr>
<tr>
<td>$U$</td>
<td>an utterance</td>
</tr>
<tr>
<td>$w$</td>
<td>a word in an utterance</td>
</tr>
<tr>
<td>$M$</td>
<td>number of utterances in $D$</td>
</tr>
<tr>
<td>$m$</td>
<td>$m$th utterance in $D$</td>
</tr>
<tr>
<td>$N$</td>
<td>length of an utterance</td>
</tr>
<tr>
<td>$V$</td>
<td>vocabulary</td>
</tr>
<tr>
<td>$e$</td>
<td>embedding mapping function</td>
</tr>
<tr>
<td>$h^{enc}_t$</td>
<td>encoder hidden state at time step $t$</td>
</tr>
<tr>
<td>$h^{dec}_t$</td>
<td>decoder hidden state at time step $t$</td>
</tr>
<tr>
<td>$h^u$</td>
<td>hidden state of the utterance encoder</td>
</tr>
<tr>
<td>$h^{con}$</td>
<td>hidden state of the context encoder</td>
</tr>
<tr>
<td>$M$</td>
<td>memory cells</td>
</tr>
<tr>
<td>$b$</td>
<td>variational memory output</td>
</tr>
<tr>
<td>$z$</td>
<td>latent variable</td>
</tr>
<tr>
<td>$P$</td>
<td>prior distribution</td>
</tr>
<tr>
<td>$Q$</td>
<td>posterior distribution</td>
</tr>
<tr>
<td>$F$</td>
<td>forget gate</td>
</tr>
<tr>
<td>$U$</td>
<td>update gate</td>
</tr>
</tbody>
</table>

Figure 1: Two different responses for the same context.
2.2 Sequence to Sequence Models

Our work is based on sequence-to-sequence (SEQ2SEQ) models. Sequence-to-sequence models prevail in natural language modeling and machine translation tasks [9]. Shang et al. [34], Sordoni et al. [38] and Vinyals and Le [41] introduced it into conversation modeling. It composed of an encoder, which takes in words sequence of previous utterances and outputs a fixed-sized context vector which summarizes all previous utterances. It also provides a decoder to generate the next utterance word-by-word based on the context vector and the recurrent hidden state through the prediction over a discrete vocabulary $V$.

The encoder and decoder process the input sequence and output sequence based on recurrent neural network (RNN). At each step, the encoder RNN unit takes in a word and updates its hidden state:

$$h_{enc}^t = \sigma(h_{enc}^{t-1}, e_w_t),$$

(2)

while decoder RNN is computed by

$$h_{dec}^t = \sigma(h_{dec}^{t-1}, e_w, c),$$

(3)

where $\sigma$ is a nonlinear activation function. It can be as simple as an element-wise logistic sigmoid function and more complex ones like long short-term memory (LSTM)[16] or Gated Recurrent Unit (GRU) [9]. $e$ is a word embedding mapping function. $c$ is the last hidden state of encoder RNN, which is the summary of the whole input sequence.

The output distribution of the decoder is used to predict the next token and is parameterized by a softmax function over an affine transformation of the decoder RNN hidden state $h_{dec}^t$:

$$P_\theta(w_{t+1} = v|w_1, ..., w_t) = \frac{\exp(g(h_{dec}^t, v))}{\sum_{v'} \exp(g(h_{dec}^t, v'))},$$

(4)

where $g$ is the affine-transformation function. The model parameters are turned by maximizing the log-likelihood over the training instances by stochastic gradient descent.

3 HIERARCHICAL VARIATIONAL MEMORY NETWORK

In this section, we propose a hierarchical variational memory network (HVMN) to model the dialogue generation process. The model employs a hierarchical RNN to model the dialogue utterances in both the utterance level and the context level. In particular, a dialogue consisting of utterance sequence $U_1, ..., U_m$ is modeled as:

$$P(U_1, ..., U_M) = \prod_{m=1}^M P(U_m|U_{<m}),$$

(5)
As each utterance $U_m$ is a sequence of words $w_{m,1}, \ldots, w_{m,N_m}$, a dialogue is then formulated as:

$$P(U_1, \ldots, U_M) = \prod_{m=1}^{M} \prod_{n=1}^{N_m} P(w_{m,n}|U_{<m}, w_{m,<n}).$$

A variational memory network then utilizes a latent variable and a memory block to capture the abstract and concrete details and complex long-term dependencies during the dialogue tracing. When decoding, the previous generated words are also given as the input to the decoder. The whole architecture is shown in Figure 2. HVMN learns to generate sequences in four steps:

1. The utterance encoder RNN encodes each turn of utterance word-by-word into a fixed-size vector, which is then sequentially given as input to the context RNN.
2. The context RNN computes a hidden state $h_{m,con}$ for $m$'th utterance.
3. Then at variational memory block, a latent variable is sampled to retrieve the memory cells as $b_m$, meanwhile the memory is also updated with $h_{m,con}$ to accumulatively memorize the current utterance.
4. The decoder RNN takes $h_{m,con}$ and $b_m$ as input to generate the response.

Next, we will first introduce the hierarchical utterance encoding process through encoder RNN and context RNN in §3.1. Then, we detail the variational memory reading and updating mechanism in §3.2. Finally, the response decoding is depicted in §3.3.

### 3.1 Utterance Encoding

The hierarchical recurrent neural network [37] regards the utterance sequence in two levels: each utterance is composed of a sequence of words, and a dialogue is a sequence of utterances.

Based on such observation, for an utterance $U_m = w_{m,1}, \ldots, w_{m,N_m}$, the utterance encoder RNN maps it to an utterance vector, which is the hidden state after $w_{m,N_m}$ is processed. A word $w_{m,n}$ is encoded by:

$$h_{u,m, n} = \sigma(h_{u,m-1, n}, e_{w_{m,n}}),$$

where $\sigma$ is the nonlinear activation function like LSTM or GRU, $e_{w_{m,n}}$ is the embedding of word $w_{m,n}$, and $h_{u,m,n}$ is the hidden state after processing $w_{m,n}$. $h_{u,m,n}$ is the utterance vector $h^u_{m,n}$ of utterance $U_m$, which can be viewed as the summarization of current utterance.

Then, the context RNN records the utterance sequence by:

$$h_{m,con} = \sigma(h_{m-1,con}, h^u_{m}),$$

where $h_{m,con}$ is the hidden state of context RNN given utterance vector $h^u_{m}$, and $\sigma$ is the same non-linear activation function in Eq.7. $h_{m,con}$ is the summary of all the observed previous utterances.

### 3.2 Variational Memory Network

**Variational Memory Reading Mechanism.**

The model draws $b_m$ from memory $M$ using a continuous stochastic latent variable $z_m$:

$$b_m = \sum_i M^i_m \otimes z^i_m,$$  

where $M_m \in \mathbb{R}^{d_m \times d}, z_m \in \mathbb{R}^{d},$ and $\otimes$ is Hadamard product function. Figure 3 illustrates the variational memory reading mechanism.

The latent variable $z_m$ is conditioned on all the previous observed tokens, namely the output of context RNN $h_{m,con}$, and is computed for each utterance:

$$P(z_m|U_{<m}) = N(\mu_{prior}(U_{<m}), \Sigma_{prior}(U_{<m})),$$

where $N(\mu, \Sigma)$ is the multivariate normal distribution with mean $\mu \in \mathbb{R}^d$ and a constrained diagonal covariance matrix $\Sigma \in \mathbb{R}^{d \times d}$. $z_m$ is inferred by maximizing the lower-bound for each utterance:

$$\log P(U_1, \ldots, U_m) \geq \sum_{m=1}^{M} -KL(Q_\psi(z_m|U_1, \ldots, U_m)||P(z_m|U_{<m}))$$

$$+ E_{Q_\psi(z_m|U_1, \ldots, U_m)}[\log P(U_m|z_m, U_{<m})].$$

where $KL(Q||P)$ is the Kullback-Leibler (KL) divergence between distributions $Q$ and $P$. The posterior distribution $Q_\psi$ approximates the intractable true posterior distribution:

$$Q_\psi(z_m|U_1, \ldots, U_m)$$

$$= N(\mu_{posterior}(U_1, \ldots, U_m), \Sigma_{posterior}(U_1, \ldots, U_m))$$

where $\mu_{posterior}$ is the approximate posterior mean, and $\Sigma_{posterior}$ is the posterior covariance diagonal matrix conditioned on previous utterances $U_1, \ldots, U_{m-1}$ and current utterance $U_m$.

The approximate prior and posterior mean and covariance are computed through a feedforward network, respectively. The former is conditioned on $h_{m,con}$ and the latter is conditioned on both $h_{m,con}$ and $h^{enc}_{m}$. A matrix multiplication is performed to compute mean, and a matrix multiplication function followed by a softplus function is implemented to output the diagonal covariance matrix:

$$e_{prior} = \mu_{prior} + \Sigma_{prior} \odot e_{prior},$$

$$e_{posterior} = \mu_{posterior} + \Sigma_{posterior} \odot e_{posterior},$$

where $e_{prior}$ and $e_{posterior}$ are standard Gaussian variables.

**Variational Memory Updating.**

Figure 4 illustrates the variational memory updating mechanism. For each utterance, the memory $M_m$ is updated with $h_{m,con}$ in order to capture details for long-term dependencies. Inspired by the writing mechanism of neural turing machines [14], we utilize a Forget and an Update operation. In particular, the forget gate $F_m \in \mathbb{R}^{d}$ defines to what extent the value of each memory cell to be erased, and the update gate $U_m \in \mathbb{R}^{d}$ specializes how much information.
of $h_{con}^m$ can be written to the memory. Typically, the memory is updated by:

$$M_m = F_m \odot M_{m-1} + U_m \odot h_{update}^m,$$  \hspace{1cm} (16)

where $F_m$ and $U_m$ are computed by a sigmoid function parameterized with a linear combination of $h_{con}^m$, $b_m$, $M_m$:

$$F_m = \text{sigmoid}(h_{con}^m, b_m, M_m)$$

$$U_m = \text{sigmoid}(h_{con}^m, b_m, M_m)$$

$h_{update}^m$ is defined as:

$$h_{update}^m = \sigma(h_{con}^m, b_m),$$  \hspace{1cm} (19)

where $\sigma$ is a non-linear activation function like sigmoid or tanh.

### 3.3 Dialogue Decoding

During dialogue generation, given the observed utterances, the latent variable $z_m$ is drawn from the prior $N(\mu, \Sigma)$. Then, $z_m$ retrieves $b_m$ from the memory cells. Finally, the embedding of last predicted word $e_{w_{t-1}}$, $b_m$, and the output of context RNN $h_{con}^m$ are given as input to the decoder RNN. In particular, the recurrent unit of decoder is defined as:

$$h_t^{dec} = \sigma(h_{t-1}^{dec}, e_{w_{t-1}}, h_{con}^m, b_m),$$  \hspace{1cm} (20)

where $\sigma$ can be a simple sigmoid function or GRU, LSTM.

The probability distribution of the next token is given by a softmax function over $h_t^{dec}$:

$$P(y|w_{t+1} = v|h_t^{dec}) = \frac{\exp(g(h_t^{dec}, v))}{\sum_{v'} \exp(g(h_t^{dec}, v'))},$$  \hspace{1cm} (21)

where $g$ is the affine-transformation function.

### 3.4 Discussions

The conditional prior distribution over variable $z_m$ injects variation for dialogue generation at the utterance-level, while the conditional distribution over word tokens performs variation at the word-level. As $z_m$ is constrained by the KL divergence between the prior and posterior, $z_m$ varies slowly along the utterances, and makes higher-level decisions about what to generate, like the conversation topic, speaker goals or sentiment of the utterance, which helps model long-term output trajectories [32]. In Serban et al. [32], $z_m$ directly guides the decoder, which increases the response diversity, however, the response appropriateness is weakened due to the lack of long-term memories. As $z_m$ is a single vector conditioned on all previous observed tokens and injected with Gaussian noise, details of the utterances are unable to be well managed.

In HVMN, the latent variable $z_m$ incorporates with the memory cells, which mimics the random access of relevant histories, where $z_m$ focuses on higher-level abstraction, like topics, sentiments, and personalities [22], in the meantime the memory cells specialize in maintaining the long-term details of the observed utterances. Memory cells are retrieved by the stochastic latent variable $z_m$, and then updated deterministically with the new utterance.

## 4 Experiments

To evaluate the effectiveness of the hierarchical variational memory neural model, we list the following questions to guide the reminder of our experiments:

- **RQ1**: Is our model effective for generating dialogues? Does it outperform state-of-the-art baselines?
- **RQ2**: How does our proposed method perform in human evaluation experiments?
- **RQ3**: What is the effect of context length for generating dialogues in our method?

Next, we introduce the datasets in §4.1. The baselines are listed in §4.2 and evaluation metrics are described in §4.3. Details of the training setting are described in §4.4.

### 4.1 Datasets

In order to assess the performance of our methods, we conduct experiments on three datasets with different styles. Two of them have been used in previous work [26, 43], and another one is extracted from JD.com. Table 2 provides descriptive statistics about our datasets.

**Ubuntu Technical Corpus**

Our first dataset, the *Ubuntu Dialogue Corpus* [26], is an English multi-turn dialogue corpus, containing about 500,000 dialogues extracted from the Ubuntu Internet Relayed Chat channel. A conversation begins with an Ubuntu-related technical problem, and typically goal driven dataset.

**Douban Conversation Corpus**

Different from the domain specific Ubuntu corpus, our second dataset, the *Douban Conversation Corpus* [43], is a Chinese multi-turn open domain conversation corpus collected from Douban groups, a popular social networking service in China. There exist multiple responses for one context. The corpus contains 0.5 million dialogues for training, more than 20,000 dialogues for validation, and 5,001 dialogues for test.

---

1https://www.jd.com

2https://www.douban.com/group
Table 2: Corpus statistics including number of dialogues in training, validation and test sets, average number of turns, utterances, words per dialogue, and vocabulary size.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>#Train</th>
<th>#Validation</th>
<th>#Test</th>
<th>#Avg. Turns</th>
<th>#Avg. Utterances</th>
<th>#Avg. Words</th>
<th>#Vocab</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ubuntu</td>
<td>448833</td>
<td>19584</td>
<td>18920</td>
<td>4.94</td>
<td>7.48</td>
<td>102.21</td>
<td>268487</td>
</tr>
<tr>
<td>Douban</td>
<td>501186</td>
<td>20000</td>
<td>5001</td>
<td>7.69</td>
<td>-</td>
<td>130.66</td>
<td>304988</td>
</tr>
<tr>
<td>JD</td>
<td>415000</td>
<td>15000</td>
<td>5005</td>
<td>11.83</td>
<td>20.43</td>
<td>266.09</td>
<td>600739</td>
</tr>
</tbody>
</table>

JD Customer Service Corpus
We collect customer service dialogues from JD.com, named the JD Customer Service Corpus. The JD customer service corpus consists of online retailing customer service dialogues. In JD corpus, each conversation is between a customer and a customer service staff. The corpus contains 415,000 dialogues for training, 15,000 dialogues for validation, and 5,005 for the test. The average number of turns and utterances in a dialogue session are much larger than the above two corpus.

4.2 Baselines
To evaluate the effectiveness of HVMN, we make comparisons between the proposed HVMN and the following state-of-the-art neural dialogue generation models in our experiments:
- **SEQ2SEQ**: It is a sequence-to-sequence dialogue model, also known as the recurrent encoder-decoder model [34, 41]. As a widely-used neural machine translation approach, SEQ2SEQ has been successfully applied to dialogue generation [41].
- **HRED**: It is a hierarchical encoder-decoder model for dialogue generation [33].
- **VIHRED**: It is a latent variable hierarchical recurrent encoder-decoder model [32].

4.3 Evaluation Methods
We divide our evaluation metrics into automatic evaluation metrics and human evaluation metrics.

**Automatic Evaluation Metrics**
Evaluating dialogue system is not a trivial problem [25]. Liu et al. [25] disclosed that word-overlap automatic metrics like BLEU [28] or ROUGE [24] are not well correlate with human evaluations regarding response quality. To evaluate the semantic relevance between the candidate response and target response, we adopt three embedding-based topic similarity metrics proposed by Liu et al. [25]: Embedding Average (Average), Embedding Extrema (Extrema) and Embedding Greedy (Greedy) [11, 27, 31]. The embedding-based metrics actually calculate the similarity between the generated response and the actual response in the embedding space, which are alternatives to word-overlap based metrics and actually take the meaning of each word into consideration.

We use the publicly available Word2Vec to train word embedding. For English, we train word embeddings on Google News Corpus, while for Chinese, the word embeddings are approximated on Chinese Giga-word corpus [13], segmented by zpar [45]. Sentence-level embedding is approximated by aggregating the individual embeddings of words in the sentence. With the sentence-level embedding, the candidate response and target response can be measured by standard similarity metrics, e.g. cosine similarity.

To evaluate the informativeness of the response (contrast with the general dull and ‘safe’ responses), we propose an average trigram word entropy metric. In particular, for a word $w_n$ in a response $U$, the trigram word entropy is defined as:

$$H(w_n) = -p(w_n|w_{n-2}, w_{n-1}) \log(p(w_n|w_{n-2}, w_{n-1})).$$

The trigram model is trained on the training set of each corpus.

**Human Evaluation**
To further validate the effectiveness of our model, we compare the responses from different models on Ubuntu corpus by human evaluations. We choose Ubuntu corpus because it is a large public available domain specific technical dataset. It is much more convenient to discriminate a better response regarding whether a technical problem is well understood and tackled. For each sequence of utterances, the responses generated by all the systems are listed in a random order. The system IDs are also anonymized. Each dialogue sequence is allocated to an evaluator randomly. Evaluators are requested to choose the responses on two dimensions: appropriateness to the context, and informativeness. They first choose a response that is more appropriate than other responses given the previous dialogue histories fro a dialogue session. Then, they select a response that is more informative and useful than other responses. By utilizing the two dimensions, we hope to discriminate whether a response is reasonable for a dialogue session and whether it is informative enough compared with a generic safe dull response. If the evaluators disagree all listed responses, or if they cannot understand the dialogue context, they can make no choice and skip.

4.4 Training Procedures
All the models are optimized using Adam [17] with a learning rate of 0.0002 and a batch size of 80. The hyperparameters and early stop patience are chosen according to the variational lower-bound. During testing, we apply beam search with five beams to generate responses. For VIHRED, we set the dimensions of the latent variable $d_i = 100$. For HVMN, a memory block $M \in \mathbb{R}^{d_i \times d_k}$ is augmented with $x_m$, $d_i$ is set to 10 and $d_k$ to 100. The baseline SEQ2SEQ employs LSTM as the recurrent unit with 1000 hidden units, while other models utilize 500 hidden units, and the dimensionality of other parameters are set accordingly. We set the vocabulary size of Ubuntu corpus as 20000; while we set the vocabulary size of both Douban and JD as 50000. For all baselines and HVMN, we apply the truncated back-propagation and the gradient clipping.

In the following subsections, we first look at the overall model performance on metric-based evaluations. Then, the human-based
### 4.5 Metric-based Evaluation Results

Table 3 lists the performance of each model. HVMN consistently outperforms almost all the baselines on three corpora, in terms of both the embedding-based topic-similarity metrics and the average trigram word entropy metric (except the word entropy metric on Ubuntu technical corpus). The improvements on Douban dataset are statistically significant (t-test with $p \leq 0.01$) on embedding average and greedy. Not surprisingly, the performance of SEQ2SEQ is the worst among all methods on all metric-based evaluations. VHRED performs better than HRED on all three topic-similarity metrics on Douban dataset, while HRED actually works better on both Ubuntu and JD dataset. This is consistent with the fact that Douban is an open domain social networking conversation dataset, which is of higher response diversity, while Ubuntu and JD are domain specific dataset. The better performance of VHRED on Douban corpus clearly verifies its ability of increasing the response diversity when comparing with HRED.

For all tested methods, their performance on Douban conversation corpus is much lower than Ubuntu technical corpus and JD e-commerce conversation corpus, while the entropies are much higher. It indicates that the dialogues in Douban corpus are more informative and much harder for dialogue generation task, which agrees with the fact that each context in Douban corpus usually consists of multiple responses. Intuitively, as introduced previously, Douban corpus is an open domain social networking conversation corpus, while Ubuntu technical corpus and JD customer service corpus are more domain specific. The response diversity differs among open domain and domain specific corpus. The higher response entropies on Douban corpus clearly verify such fact.

<table>
<thead>
<tr>
<th>Model</th>
<th>Average</th>
<th>Greedy</th>
<th>Extrema</th>
<th>H(w)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ubuntu</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEQ2SEQ</td>
<td>0.215603</td>
<td>0.168833</td>
<td>0.126480</td>
<td>0.2638</td>
</tr>
<tr>
<td>HRED</td>
<td>0.541548</td>
<td>0.411681</td>
<td>0.319299</td>
<td>0.3082</td>
</tr>
<tr>
<td>VHRED</td>
<td>0.534103</td>
<td>0.402670</td>
<td>0.306242</td>
<td>0.2878</td>
</tr>
<tr>
<td>HVMN</td>
<td><strong>0.558392</strong></td>
<td><strong>0.422914</strong></td>
<td><strong>0.322032</strong></td>
<td><strong>0.3002</strong></td>
</tr>
<tr>
<td><strong>Douban</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEQ2SEQ</td>
<td>0.024255</td>
<td>0.002961</td>
<td>0.023805</td>
<td>1.2253</td>
</tr>
<tr>
<td>HRED</td>
<td>0.030904</td>
<td>0.003817</td>
<td>0.029889</td>
<td>1.5116</td>
</tr>
<tr>
<td>VHRED</td>
<td>0.042774</td>
<td>0.005147</td>
<td>0.041703</td>
<td>1.3671</td>
</tr>
<tr>
<td>HVMN</td>
<td><strong>0.053293</strong></td>
<td><strong>0.006507</strong></td>
<td><strong>0.051560</strong></td>
<td><strong>3.1042</strong></td>
</tr>
<tr>
<td><strong>JD</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEQ2SEQ</td>
<td>0.309752</td>
<td>0.204973</td>
<td>0.279654</td>
<td>0.3219</td>
</tr>
<tr>
<td>HRED</td>
<td>0.737606</td>
<td>0.500789</td>
<td>0.675900</td>
<td>0.3286</td>
</tr>
<tr>
<td>VHRED</td>
<td>0.609605</td>
<td>0.413422</td>
<td>0.558891</td>
<td>0.3473</td>
</tr>
<tr>
<td>HVMN</td>
<td><strong>0.752574</strong></td>
<td><strong>0.511170</strong></td>
<td><strong>0.691818</strong></td>
<td><strong>0.3555</strong></td>
</tr>
</tbody>
</table>

Table 3: Evaluations on embedding-based metrics. **“”** denotes significantly better than VHRED with $p \leq 0.01$.
Table 4: Average differences on human evaluation.

<table>
<thead>
<tr>
<th>Comparison</th>
<th>Appropriateness(%)</th>
<th>Informativeness(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HVMN vs SEQ2SEQ</td>
<td>77.07 : 22.93</td>
<td>84.46 : 15.54</td>
</tr>
<tr>
<td>HVMN vs HRED</td>
<td>48.52 : 51.48</td>
<td>58.41 : 41.59</td>
</tr>
<tr>
<td>VHRED vs HRED</td>
<td>44.57 : 55.43</td>
<td>52.91 : 47.09</td>
</tr>
<tr>
<td>HVMN vs VHRED</td>
<td>53.97 : 46.03</td>
<td>55.56 : 44.44</td>
</tr>
</tbody>
</table>

4.6 Effect of Context Length

Figure 5 lists the number of contexts of Ubuntu corpus. Most contexts are of lengths 0-50 and 50-100. Figures 6-8 show the performance on different lengths of contexts for three embedding-based evaluation metrics on Ubuntu corpus. Note that we do not show results in other datasets since we have similar observations. We can see that HVMN performs better than other baselines at most different lengths, while HRED perform the second best. Comparing with HRED, for the contexts with length less than 50, HVMN performs noticeable better on Embedding Average and Embedding Greedy metrics. One possible reason is that, it is harder for short contexts to generate reasonable responses than regular ones, due to the lack of effective information. Through manually examining the results, we observed that baseline models even fail to make meaningful decisions when predicting the response word-by-word, while with the variational memory, HVMN is capable of injecting variation for short contexts. When we looking at the main baseline VHRED, the performance gap between HVMN and VHRED increases when the contexts become longer, which demonstrates that HVMN works better for long contexts than VHRED. It confirms that variational memory in HVMN is able to provide more power for long term dependencies.

4.7 Human Evaluation Performance

To further validate the performance, we conduct the human evaluation on Ubuntu dataset. We choose Ubuntu corpus, because it is easy to specify whether a response is appropriate, and whether a response is informative enough in technical problem discussing.

Table 4 lists the comparison results. Line 1 and line 4 suggest that HVMN outperforms its counterparts, namely 77.07 : 22.93 and 53.97 : 46.03 on appropriateness, and 84.46 : 15.54 and 58.41 : 44.44 on informativeness. We also notice that VHRED works better than HRED in terms of informativeness (line 3, 52.91 : 47.09) and performs worse than HRED with respect to appropriateness (44.57 : 55.43). This suggests that, although Ubuntu is a domain specific technical dataset, VHRED still improves response informativeness with a bit loss of appropriateness and HRED tends to make generic safe and meaningless responses. A straightforward observation is that a more informative and longer response tends to make more mistakes comparing with a short and dull response. Therefore, the appropriateness of VHRED decreases compared with HRED, while the informativeness increases a lot.

In line 2, HVMN achieves comparable performance on appropriateness compared with HRED (48.52 : 51.48). However, HVMN remarkably outperforms HRED with respect to informativeness (58.41 : 41.59). This suggests that HVMN maintains its appropriateness while increasing the informativeness. By utilizing the variational memory, HVMN performs better at generate a diversified and "right" response in contrast with VHRED.

4.8 Qualitative Evaluation

Table 5 presents examples of generated responses for different models. It is clear that HVMN performs better at understanding contexts. The responses of HVMN are not only more appropriate but also more informative and useful.

Comparing LSTM and HRED, both systems tend to make generic, "safe" and dull response. LSTM makes short and meaningless responses for all the cases, while HRED outputs longer responses and performs slightly better and is more likely to admit that it does not understand the discussion and fails to give detailed solutions. Comparing HRED and VHRED, line 3 and 5 in Table 5 indicate that VHRED tends to output more specific answers. It gives an appropriate Linux command for line 3, however, it fails to generate a proper and effective response for line 5. Generally, HVMN doesn’t make a generic "yes/no" response but provides detailed and appropriate operations, especially for line 1 and line 4.

5 RELATED WORK

Our related work can be classified into two categories: variational neural model and memory network.

5.1 Variational Neural Model


Unlike previous models, which directly utilize the latent variable to guide the generation, the latent variable in our model is used as a key to read from the memory block and learn to make abstract high-level decisions during dialogue tracing.

5.2 Memory Network

Memory Network model is first proposed by Sukhbaatar et al. [40] and Sukhbaatar et al. [39]. Cheng et al. [8] equip the machine reader with a memory tape, which enables the model to read all the previous hidden state directly. Wang et al. [42] use an interactive memory...
Table 5: Examples of dialogues generated by various models on Ubuntu corpus. Column 1 lists the context of a dialogue, whereas Column 2 lists the generated response by different models. “→” denotes a turn exchange.

<table>
<thead>
<tr>
<th>Contexts</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>hello , I am using ubuntu 12.04 and have black screen It seems that I have gugraphical system X logged as root user , but not as my user . Can someone to end up my problem ? I have ubuntu 12.04 → are you logged on ? and do you have a text prompt ?</td>
<td>HVMN: yes , I can log in as root , but I am not able to log in as root</td>
</tr>
<tr>
<td>hey what command would I use to extract a gzip file tar . gzip → in which part ? document , assignment , labs ? → sorry labs → wait a second , my connection gets slow connection → oh sorry . can you not see it ? do you know if I have to extract that . tar . gzip → not yet finish .D skydrive was suck → lol its the only option I had . i could just send you my onenote linke → okey just for your link I get slower connections :D</td>
<td>LSTM: “unknown”</td>
</tr>
<tr>
<td>I've been told that du doesn’t count hidden files/directories size toward their parent directory size , yet in practice it seems du still does it → well read the manual . because it always did and always will ?</td>
<td>HRED: yes</td>
</tr>
<tr>
<td>Hello . I want to access a ubuntu desktop from windows 7 to know what the client is doing . How can i do that ? Hello . I want to access a ubuntu desktop from windows 7 to know what the client is doing . How can i do that ? → Try Teamviewer → I want to do without the knowledge of client</td>
<td>LHRED: yes</td>
</tr>
<tr>
<td>Hi there . I'm doing an rsync on a server . i've just looked at it remotely and its showing as 3 occurrences of it running (via ps aux</td>
<td>grep rsync ) is that normal ? → Do they all look exactly the same or is there any destinguishable be difference → all look exactly the same : s what thats mean then ?</td>
</tr>
<tr>
<td>Hello . I want to access a ubuntu desktop from windows 7 to know what the client is doing . How can i do that ? → Try Teamviewer → I want to do without the knowledge of client</td>
<td>HRED: I know , but I don’t know how to do that . I just want to know if there is a way to do it</td>
</tr>
<tr>
<td>Hello . I want to access a ubuntu desktop from windows 7 to know what the client is doing . How can i do that ? → Try Teamviewer → I want to do without the knowledge of client</td>
<td>VHRED: df -h returns nothing ?</td>
</tr>
<tr>
<td>Hello . I want to access a ubuntu desktop from windows 7 to know what the client is doing . How can i do that ? → Try Teamviewer → I want to do without the knowledge of client</td>
<td>HVMN: You’ll need to set up a VNC server on the Ubuntu machine .</td>
</tr>
<tr>
<td>hello , I am using ubuntu 12.04 and have black screen It seems that I have gugraphical system X logged as root user , but not as my user . Can someone to end up my problem ? I have ubuntu 12.04 → are you logged on ? and do you have a text prompt ?</td>
<td>LSTM: ?</td>
</tr>
<tr>
<td>hey what command would I use to extract a gzip file tar . gzip → in which part ? document , assignment , labs ? → sorry labs → wait a second , my connection gets slow connection → oh sorry . can you not see it ? do you know if I have to extract that . tar . gzip → not yet finish .D skydrive was suck → lol its the only option I had . i could just send you my onenote linke → okey just for your link I get slower connections :D</td>
<td>HRED: I know , but I don’t know how to do that . I just want to know if there is a way to do it</td>
</tr>
<tr>
<td>I've been told that du doesn’t count hidden files/directories size toward their parent directory size , yet in practice it seems du still does it → well read the manual . because it always did and always will ?</td>
<td>VHRED: df -h returns nothing ?</td>
</tr>
<tr>
<td>Hello . I want to access a ubuntu desktop from windows 7 to know what the client is doing . How can i do that ? Hello . I want to access a ubuntu desktop from windows 7 to know what the client is doing . How can i do that ? → Try Teamviewer → I want to do without the knowledge of client</td>
<td>LSTM: “unknown”</td>
</tr>
<tr>
<td>Hi there . I’m doing an rsync on a server . i’ve just looked at it remotely and its showing as 3 occurrences of it running (via ps aux</td>
<td>grep rsync ) is that normal ? → Do they all look exactly the same or is there any destinguishable be difference → all look exactly the same : s what thats mean then ?</td>
</tr>
<tr>
<td>Hello . I want to access a ubuntu desktop from windows 7 to know what the client is doing . How can i do that ? → Try Teamviewer → I want to do without the knowledge of client</td>
<td>VHRED: grep “ unknown” . “ unknown” . “ unknown” .</td>
</tr>
</tbody>
</table>

6 CONCLUSION

In this paper, we have studied the dialogue generation problem and have identified the main challenges: the long-term dependency and the informativeness. To tackle these problems, we have utilized the hierarchical structure, together with the variational memory to enhance the utterance modeling for dialogues generation. The two-level hierarchical structure naturally encodes the dialogue utterances within an utterance and across the utterance sequence. The variational memory tracks the high level abstraction, memorize the long-term details of the observed utterances, and randomly access the dialogue histories. Extensive experiments conducted on two benchmark datasets and a real-world e-commerce dataset have verified the effectiveness of our proposed method by showing significant improvements over multiple baselines in terms of metric-based evaluations and human evaluations.

Note that, our model is not only limited to dialogue generation task, it can also be applied to other tasks like machine reading and summarization. We would like to make further studies in our future work. Also, we would like to integrate external knowledge base into the memory block to make the dialogue generation capable of handling enormous structured knowledge.

ACKNOWLEDGEMENTS

The authors wish to thank the anonymous reviewers for their helpful comments. The authors wish to thank Xi Xiong for preprocessing the JD customer service corpus. Jiliang Tang is supported by the National Science Foundation (NSF) under grant number IIS-1714741 and IIS-1715940.
REFERENCES


