A Hierarchical Latent Variable Encoder-Decoder Model for Generating Dialogues

Iulian V. Serban[*], Alessandro Sordoni [‡], Ryan Lowe[◇], Laurent Charlin[◇], Joelle Pineau[◇],
Aaron Courville[*] and Yoshua Bengio[*]

[*]Department of Computer Science and Operations Research, Université de Montréal, Montreal, Canada
{iulian.vlad.serban,aaron.courville,yoshua.bengio}@umontreal.ca

[‡]Maluuba Inc, Montreal, Canada
{alessandro.sordoni}@maluuba.com

[◇]School of Computer Science, McGill University, Montreal, Canada
{ryan.lowe,lcharlin,jpineau}@cs.mcgill.ca

Abstract

Sequential data often possesses a hierarchical structure with complex dependencies between subsequences, such as found between the utterances in a dialogue. In an effort to model this kind of generative process, we propose a neural network-based generative architecture, with latent stochastic variables that span a variable number of time steps. We apply the proposed model to the task of dialogue response generation and compare it with recent neural network architectures. We evaluate the model performance through automatic evaluation metrics and by carrying out a human evaluation. The experiments demonstrate that our model improves upon recently proposed models and that the latent variables facilitate the generation of long outputs and maintain the context.

1 Introduction

Deep recurrent neural networks (RNNs) have recently demonstrated impressive results on a number of difficult machine learning problems involving the generation of sequential structured outputs [9], including language modelling [10, 18] machine translation [28, 5], dialogue [27, 24] and speech recognition [9].

While these advances are impressive, the underlying RNNs tend to have a fairly simple structure, in the sense that the only variability or stochasticity in the model occurs when an output is sampled. This is often an inappropriate place to inject variability [2, 6, 1]. This is especially true for sequential data such as speech and natural language that possess a hierarchical generation process with complex intra-sequence dependencies. For instance, natural language dialogue involves at least two levels of structure; within a single utterance the structure is dominated by local statistics of the language, while across utterances there is a distinct source of uncertainty (or variance) characterized by aspects such as conversation topic, speaker goals and speaker style.

In this paper we introduce a novel hierarchical stochastic latent variable neural network architecture to explicitly model generative processes that possess multiple levels of variability. We evaluate the proposed model on the task of dialogue response generation and compare it with recent neural network architectures. We evaluate the model qualitatively through manual inspection, and quantitatively using a human evaluation on Amazon Mechanical Turk and using automatic evaluation metrics. The

This work was carried out while A.S. was at Université de Montréal. Y.B. is a CIFAR senior Fellow.

results demonstrate that the model improves upon recently proposed models. In particular, the results highlight that the latent variables help to both facilitate the generation of long utterances with more information content, and to maintain the dialogue context.

2 Technical Background

2.1 Recurrent Neural Network Language Model

A recurrent neural network (RNN), with parameters θ, models a variable-length sequence of tokens $(w_1, \ldots, w_M)$ by decomposing the probability distribution over outputs:

$$P_\theta(w_1, \ldots, w_M) = \prod_{m=2}^{M} P_\theta(w_m \mid w_1, \ldots, w_{m-1}) P(w_1). \tag{1}$$

The model processes each observation recursively. At each time step, the model observes an element and updates its internal hidden state, $h_m = f(h_{m-1}, w_m)$, where f is a parametrized non-linear function, such as the hyperbolic tangent, the LSTM gating unit [12] or the GRU gating unit [5].[1] The hidden state acts as a sufficient statistic, which summarizes the past sequence and parametrizes the output distribution of the model: $P_\theta(w_{m+1} \mid w_1, \ldots, w_m) = P_\theta(w_{m+1} \mid h_m)$. We assume the outputs lie within a discrete vocabulary V. Under this assumption, the RNN Language Model (RNNLM) [18], the simplest possible generative RNN for discrete sequences, parametrizes the output distribution using the softmax function applied to an affine transformation of the hidden state h_m. The model parameters are learned by maximizing the training log-likelihood using gradient descent.

2.2 Hierarchical Recurrent Encoder-Decoder

The hierarchical recurrent encoder-decoder model (HRED) [26, 24] is an extension of the RNNLM. It extends the encoder-decoder architecture [5] to the natural dialogue setting. The HRED assumes that each output sequence can be modelled in a two-level hierarchy: sequences of sub-sequences, and sub-sequences of tokens. For example, a dialogue may be modelled as a sequence of utterances (sub-sequences), with each utterance modelled as a sequence of words. Similarly, a natural-language document may be modelled as a sequence of sentences (sub-sequences), with each sentence modelled as a sequence of words. The HRED model consists of three RNN modules: an *encoder* RNN, a *context* RNN and a *decoder* RNN. Each sub-sequence of tokens is deterministically encoded into a real-valued vector by the *encoder* RNN. This is given as input to the *context* RNN, which updates its internal hidden state to reflect all information up to that point in time. The *context* RNN deterministically outputs a real-valued vector, which the *decoder* RNN conditions on to generate the next sub-sequence of tokens. For additional details see [26, 24].

2.3 A Deficient Generation Process

In the recent literature, it has been observed that the RNNLM and HRED, and similar models based on RNN architectures, have critical problems generating meaningful dialogue utterances [24, 15]. We believe that the root cause of these problems arise from the parametrization of the output distribution in the RNNLM and HRED, which imposes a strong constraint on the generation process: the only source of variation is modelled through the conditional output distribution. This is detrimental from two perspectives: from a probabilistic perspective, with stochastic variations injected only at the low level, the model is encouraged to capture local structure in the sequence, rather than global or long-term structure. This is because random variations injected at the lower level are strongly constrained to be in line with the immediate previous observations, but only weakly constrained to be in line with older observations or with future observations. One can think of random variations as injected via i.i.d. noise variables, added to deterministic components, for example. If this noise is injected at a higher level of representation, spanning longer parts of the sequence, its effects could correspond to longer-term dependencies. Second, from a computational learning perspective, the state h_m of the RNNLM (or *decoder* RNN of HRED) has to summarize all the past information up to time step m in order to (a) generate a probable next token (short term goal) and simultaneously (b) to occupy a position in embedding space which sustains a realistic output trajectory, in order to generate

[1] We concatenate the LSTM cell and cell input hidden states into a single state h_m for notational simplicity.

probable future tokens (long term goal). Due to the vanishing gradient effect, shorter-term goals will have more influence: finding a compromise between these two disparate forces will likely lead the training procedure to model parameters that focus too much on predicting only the next output token. In particular for high-entropy sequences, the models are very likely to favour short-term predictions as opposed to long-term predictions, because it is easier to only learn h_m for predicting the next token compared to sustaining a long-term trajectory, which at every time step is perturbed by a highly noisy source (the observed token).

3 Latent Variable Hierarchical Recurrent Encoder-Decoder (VHRED)

Motived by the previous discussion, we now introduce the latent variable hierarchical recurrent encoder-decoder (VHRED) model. This model augments the HRED model with a latent variable at the decoder, which is trained by maximizing a variational lower-bound on the log-likelihood. This allows it to model hierarchically-structured sequences in a two-step generation process—first sampling the latent variable, and then generating the output sequence—while maintaining long-term context.

Let $\mathbf{w}_1, \dots, \mathbf{w}_N$ be a sequence consisting of N sub-sequences, where $\mathbf{w}_n = (w_{n,1}, \dots, w_{n,M_n})$ is the n'th sub-sequence and $w_{n,m} \in V$ is the m'th discrete token in that sequence. The VHRED model uses a stochastic latent variable $\mathbf{z}_n \in \mathbb{R}^{d_z}$ for each sub-sequence $n = 1, \dots, N$ conditioned on all previous observed tokens. Given $\mathbf{z}_n$, the model next generates the n'th sub-sequence tokens $\mathbf{w}_n = (w_{n,1}, \dots, w_{n,M_n})$:

$$P_\theta(\mathbf{z}_n \mid \mathbf{w}_1, \dots, \mathbf{w}_{n-1}) = \mathcal{N}(\boldsymbol{\mu}_{\text{prior}}(\mathbf{w}_1, \dots, \mathbf{w}_{n-1}), \Sigma_{\text{prior}}(\mathbf{w}_1, \dots, \mathbf{w}_{n-1})), \tag{2}$$

$$P_\theta(\mathbf{w}_n \mid \mathbf{z}_n, \mathbf{w}_1, \dots, \mathbf{w}_{n-1}) = \prod_{m=1}^{M_n} P_\theta(w_{n,m} \mid \mathbf{z}_n, \mathbf{w}_1, \dots, \mathbf{w}_{n-1}, w_{n,1}, \dots, w_{n,m-1}), \tag{3}$$

where $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ is the multivariate normal distribution with mean $\boldsymbol{\mu} \in \mathbb{R}^{d_z}$ and covariance matrix $\Sigma \in \mathbb{R}^{d_z \times d_z}$, which is constrained to be a diagonal matrix.

The VHRED model (figure 1) contains the same three components as the HRED model. The *encoder* RNN deterministically encodes a single sub-sequence into a fixed-size real-valued vector. The *context* RNN deterministically takes as input the output of the *encoder* RNN, and encodes all previous sub-sequences into a fixed-size real-valued vector. This vector is fed into a two-layer feed-forward neural network with hyperbolic tangent gating function. A matrix multiplication is applied to the output of the feed-forward network, which defines the multivariate normal mean $\boldsymbol{\mu}_{\text{prior}}$. Similarly, for the diagonal covariance matrix Σ_{prior} a different matrix multiplication is applied to the net's output followed by softplus function, to ensure positiveness [6].

The model's latent variables are inferred by maximizing the variational lower-bound, which factorizes into independent terms for each sub-sequence:

$$\begin{aligned} \log P_\theta(\mathbf{w}_1, \dots, \mathbf{w}_N) \geq \sum_{n=1}^{N} & -\text{KL}\left[Q_\psi(\mathbf{z}_n \mid \mathbf{w}_1, \dots, \mathbf{w}_n) || P_\theta(\mathbf{z}_n \mid \mathbf{w}_1, \dots, \mathbf{w}_{n-1})\right] \\ & + \mathbb{E}_{Q_\psi(\mathbf{z}_n \mid \mathbf{w}_1, \dots, \mathbf{w}_n)}\left[\log P_\theta(\mathbf{w}_n \mid \mathbf{z}_n, \mathbf{w}_1, \dots, \mathbf{w}_{n-1})\right], \end{aligned} \tag{4}$$

where KL$[Q||P]$ is the Kullback-Leibler (KL) divergence between distributions Q and P. The distribution $Q_\psi(\mathbf{z} \mid w_1, \dots, w_M)$ is the approximate posterior distribution (also known as the *encoder model* or *recognition model*), which aims to approximate the intractable true posterior distribution:

$$\begin{aligned} Q_\psi(\mathbf{z}_n \mid \mathbf{w}_1, \dots, \mathbf{w}_N) = Q_\psi(\mathbf{z}_n \mid \mathbf{w}_1, \dots, \mathbf{w}_n) &= \mathcal{N}(\boldsymbol{\mu}_{\text{posterior}}(\mathbf{w}_1, \dots, \mathbf{w}_n), \Sigma_{\text{posterior}}(\mathbf{w}_1, \dots, \mathbf{w}_n)) \\ &\approx P_\psi(\mathbf{z}_n \mid \mathbf{w}_1, \dots, \mathbf{w}_N), \end{aligned} \tag{5}$$

where $\boldsymbol{\mu}_{\text{posterior}}$ defines the approximate posterior mean and $\Sigma_{\text{posterior}}$ defines the approximate posterior covariance matrix (assumed diagonal) as a function of the previous sub-sequences $\mathbf{w}_1, \dots, \mathbf{w}_{n-1}$ and the current sub-sequence $\mathbf{w}_n$. The posterior mean $\boldsymbol{\mu}_{\text{posterior}}$ and covariance $\Sigma_{\text{posterior}}$ are determined in the same way as the prior, via a matrix multiplication with the output of the feed-forward network, and with a softplus function applied for the covariance.

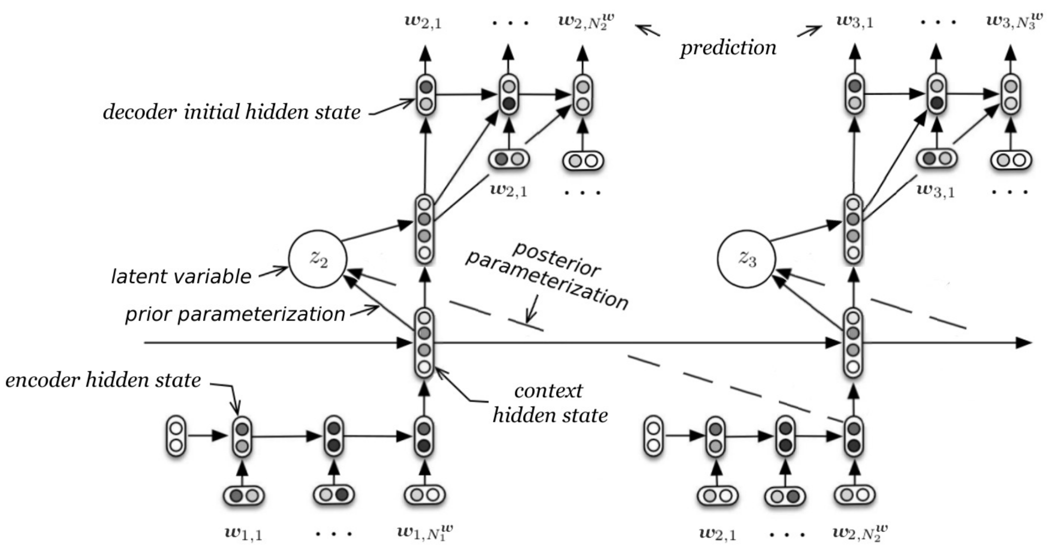

$w_{2,1}$ . . . w_{2,N_2^w}
prediction
$w_{3,1}$ . . . w_{3,N_3^w}
decoder initial hidden state
$w_{2,1}$ . . .
$w_{3,1}$ . . .
z_2
z_3
latent variable
posterior parameterization
prior parameterization
encoder hidden state
context hidden state
$w_{1,1}$ . . . w_{1,N_1^w}
$w_{2,1}$ . . . w_{2,N_2^w}

Figure 1: Computational graph for VHRED model. Rounded boxes represent (deterministic) real-valued vectors. Variables $\mathbf{z}$ represent latent stochastic variables.

At test time, conditioned on the previous observed sub-sequences $(\mathbf{w}_1, \dots, \mathbf{w}_{n-1})$, a sample $\mathbf{z}_n$ is drawn from the prior $\mathcal{N}(\boldsymbol{\mu}_{\text{prior}}(\mathbf{w}_1, \dots, \mathbf{w}_{n-1}), \Sigma_{\text{prior}}(\mathbf{w}_1, \dots, \mathbf{w}_{n-1}))$ for each sub-sequence. This sample is concatenated with the output of the *context* RNN and given as input to the *decoder* RNN as in the HRED model, which then generates the sub-sequence token-by-token. At training time, for $n = 1, \dots, N$, a sample $\mathbf{z}_n$ is drawn from the approximate posterior $\mathcal{N}(\boldsymbol{\mu}_{\text{posterior}}(\mathbf{w}_1, \dots, \mathbf{w}_n), \Sigma_{\text{posterior}}(\mathbf{w}_1, \dots, \mathbf{w}_n))$ and used to estimate the gradient of the variational lower-bound given by Eq. (4). The approximate posterior is parametrized by its own one-layer feed-forward neural network, which takes as input the output of the *context* RNN at the current time step, as well as the output of the *encoder* RNN for the next sub-sequence.

The VHRED model greatly helps to reduce the problems with the generation process used by the RNNLM and HRED model outlined above. The variation of the output sequence is now modelled in two ways: at the sequence-level with the conditional prior distribution over $\mathbf{z}$, and at the sub-sequence-level (token-level) with the conditional distribution over tokens $w_1, \dots, w_M$. The variable $\mathbf{z}$ helps model long-term output trajectories, by representing high-level information about the sequence, which in turn allows the variable h_m to primarily focus on summarizing the information up to token M. Intuitively, the randomness injected by the variable $\mathbf{z}$ corresponds to higher-level decisions, like topic or sentiment of the sentence.

4 Experimental Evaluation

We consider the problem of conditional natural language response generation for dialogue. This is an interesting problem with applications in areas such as customer service, technical support, language learning and entertainment [29]. It is also a task domain that requires learning to generate sequences with complex structures while taking into account long-term context [17, 27].

We consider two tasks. For each task, the model is given a dialogue context, consisting of one or more utterances, and the goal of the model is to generate an appropriate next response to the dialogue. We first perform experiments on a **Twitter Dialogue Corpus** [22]. The task is to generate utterances to append to existing Twitter conversations. The dataset is extracted using a procedure similar to Ritter et al. [22], and is split into training, validation and test sets, containing respectively $749,060$, $93,633$ and $10,000$ dialogues. Each dialogue contains 6.27 utterances and 94.16 tokens on average.[2] The dialogues are fairly long compared to recent large-scale language modelling corpora, such as the 1 Billion Word Language Model Benchmark [4], which focus on modelling single sentences.

[2] Due to Twitter's terms of service we are not allowed to redistribute Twitter content. Therefore, only the tweet IDs can be made public. These are available at: `www.iulianserban.com/Files/TweetIDs.zip`.

We also experiment on the **Ubuntu Dialogue Corpus** [17], which contains about $500,000$ dialogues extracted from the *#Ubuntu* Internet Relayed Chat channel. Users enter the chat channel with a Ubuntu-related technical problem, and other users try to help them. For further details see Appendix 6.1.[3] We chose these corpora because they are large, and have different purposes—Ubuntu dialogues are typically goal driven, where as Twitter dialogues typically contain social interaction ("chit-chat").

4.1 Training and Evaluation Procedures

We optimize all models using Adam [13]. We choose our hyperparameters and early stop with patience using the variational lower-bound [9]. At test time, we use beam search with 5 beams for outputting responses with the RNN decoders [10]. For the VHRED models, we sample the latent variable $\mathbf{z}_n$, and condition on it when executing beam search with the RNN decoder. For Ubuntu we use word embedding dimensionality of size 300, and for Twitter we use word embedding dimensionality of size 400. All models were trained with a learning rate of 0.0001 or 0.0002 and with mini-batches containing 40 or 80 training examples. We use a variant of truncated back-propagation and we apply gradient clipping. Further details are given in Appendix 6.2.

Baselines On both Twitter and Ubuntu we compare to an LSTM model of 2000 hidden units. On Ubuntu, the HRED model has 500, 1000 and 500 hidden units for the *encoder*, *context* and *decoder* RNNs respectively. The *encoder* RNN is a standard GRU RNN. On Twitter, the HRED model *encoder* RNN is a bidirectional GRU RNN encoder, where the forward and backward RNNs each have 1000 hidden units, and *context* RNN and *decoder* RNN have each 1000 hidden units. For reference, we also include a non-neural network baseline, specifically the TF-IDF retrieval-based model proposed in [17].

VHRED The *encoder* and *context* RNNs for the VHRED model are parametrized in the same way as the corresponding HRED models. The only difference in the parametrization of the *decoder* RNN is that the *context* RNN output vector is now concatenated with the generated stochastic latent variable. Furthermore, we initialize the feed-forward networks of the prior and posterior distributions with values drawn from a zero-mean normal distribution with variance 0.01 and with biases equal to zero. We also multiply the diagonal covariance matrices of the prior and posterior distributions with 0.1 to make training more stable, because a high variance makes the gradients w.r.t. the reconstruction cost unreliable, which is fatal at the beginning of the training process.

The VHRED's *encoder* and *context* RNNs are initialized to the parameters of the corresponding converged HRED models. We also use two heuristics proposed by Bowman et al. [3]: we drop words in the decoder with a fixed drop rate of 25% and multiply the KL terms in eq. (4) by a scalar, which starts at zero and linearly increases to 1 over the first $60,000$ and $75,000$ training batches on Twitter and Ubuntu respectively. Applying these heuristics helped substantially to stabilize the training process and make the model use the stochastic latent variables. We experimented with the batch normalization training procedure for the feed-forward neural networks. but found that this made training very unstable without any substantial gains in performance w.r.t. the variational bound.

Evaluation Accurate evaluation of dialogue system responses is a difficult problem [8, 20]. Inspired by metrics for machine translation and information retrieval, researchers have begun adopting word-overlap metrics, however Liu et al. [16] show that such metrics have little correlation with human evaluations of response quality. We therefore carry out a human evaluation to compare responses from the different models. We also compute several statistics and automatic metrics on model responses to characterize differences between the model-generated responses.

We carry out the human study for the Twitter Dialogue Corpus on Amazon Mechanical Turk (AMT). We do not conduct AMT experiments on Ubuntu as evaluating these responses usually requires technical expertise, which is not prevalent among AMT users. We set up the evaluation study as a series of pairwise comparison experiments.[4] We show human evaluators a dialogue context along with two potential responses, one generated from each model (conditioned on dialogue context). We ask participants to choose the response most appropriate to the dialogue context. If the evaluators are indifferent to either of the two responses, or if they cannot understand the dialogue context, they can choose neither response. For each pair of models we conduct two experiments: one where the

[3] The pre-processed Ubuntu Dialogue Corpus used is available at `www.iulianserban.com/Files/UbuntuDialogueCorpus.zip`.

[4] Source code for the AMT experiments will be released upon publication.

Table 1: Wins, losses and ties (in %) of the VHRED model against the baselines based on the human study on Twitter (mean preferences ± 90% confidence intervals)

	Short Contexts			Long Contexts		
Opponent	**Wins**	**Losses**	**Ties**	**Wins**	**Losses**	**Ties**
VHRED vs LSTM	32.3 ± 2.4	$\mathbf{42.5 \pm 2.6}$	25.2 ± 2.3	$\mathbf{41.9 \pm 2.2}$	36.8 ± 2.2	21.3 ± 1.9
VHRED vs HRED	$\mathbf{42.0 \pm 2.8}$	31.9 ± 2.6	26.2 ± 2.5	$\mathbf{41.5 \pm 2.8}$	29.4 ± 2.6	29.1 ± 2.6
VHRED vs TF-IDF	$\mathbf{51.6 \pm 3.3}$	17.9 ± 2.5	30.4 ± 3.0	$\mathbf{47.9 \pm 3.4}$	11.7 ± 2.2	40.3 ± 3.4

Table 2: Twitter examples for the neural network models. The → token indicates a change of turn.

Context	**Response**
jamaica blood!! <url> → lol. yes [name]!. [name]! → what's poppin in toronto homie	**VHRED:** nothing much. just chillin. what's good with you? **LSTM:** something!!!!!!! **HRED:** something!!!!!!
whhhhhhyyyy is it suddenly cold in tallahassee today?? this is florida! how can i dream of the beach when my fingers are icey cold?? → sittin on deck in sun looking at lake travis in austin! enjoying day b4 birthday	**VHRED:** the sun is shining!!! **LSTM:** i know!!!!!!!!! **HRED:** something!!!!!!!
reaches for [name]! → - approaches him warily - hey there buddy. - swings him into the air - how you doin? → tilts his head and your wariness and reaches for your face [name].	**VHRED:** - looks up at you and smiles - you're so cute. - kisses your cheek - **LSTM:** - smiles - i'm fine thank you. how are you? **HRED:** - grins, looking at you as i look at you -
i think i want an ipad2. but i already have an ibook. want vs. need. → if you have an ibook. then you deserve an ipad!	**VHRED:** that's what i'm thinking. i'm going to buy a new one. **LSTM:** i want one!!!! **HRED:** i don't know what to do with it. i don't know what to do.
ah echt? dan anrtwoord ik volgende keer via twitter als je me smst. → haha doe maar p. goed bezig hoor myrthe d → goed bezig? die snap ik even niet p	**VHRED:** haha, ik heb er geen zin in de something, maar ik heb er geen something meer something **LSTM:** haha something something something **HRED:** something something something something something something something something
i love katie waissel more than like, anything in life. → i wanna c the show again! i need <number> meet her again! n i want a b-day tweet nxt week doubt that will happen x	**VHRED:** me too. i'm going to have to wait for her to come back. i'm so excited. **LSTM:** i'm sure she will. i'm sure she'll be fine. **HRED:** i'm sure she'll be fine, i'm sure she'll be fine.
my math teacher over here talkn bout nickels nd i'm tryna make big money i need to be in calculus → lol	**VHRED:** i need to go to the gym and get my hair done lol **LSTM:** lol **HRED:** how u been

example contexts contain at least 80 unique tokens (*long context*), and one where they contain at least 20 (not necessarily unique) tokens (*short context*). This helps compare how well each model can integrate the dialogue context into its response, since it has previously been hypothesized that for long contexts hierarchical RNNs models fare better [24, 26]. Screenshots and further details of the experiments are in Appendix 6.4.

4.2 Results of Human Evaluation

The results (table 1) show that VHRED is clearly preferred in the majority of the experiments. In particular, VHRED is strongly preferred over the HRED and TF-IDF baseline models for both short and long context settings. VHRED is also preferred over the LSTM baseline model for long contexts; however, the LSTM is preferred over VHRED for short contexts. We believe this is because the LSTM baseline tends to output much more *generic* responses (see table 4); since it doesn't model the hierarchical input structure, the LSTM model has a shorter memory span, and thus must output a response based primarily on the end of the last utterance. Such 'safe' responses are reasonable for a wider range of contexts, meaning that human evaluators are more likely to rate them as appropriate. However, we argue that a model that only outputs generic responses is undesirable for dialogue, as this leads to uninteresting and less engaging conversations. Conversely, the VHRED model is explicitly designed for long contexts, and to output a diverse set of responses due to the sampling of the latent variable. Thus, the VHRED model generates longer sentences with more semantic content than the LSTM model (see tables 3-4). This can be 'riskier' as longer utterances are more likely to contain small mistakes, which can lead to lower human preference for a single utterance. However, we believe that response diversity is crucial to maintaining interesting conversations — in the dialogue literature, generic responses are used primarily as 'back-off' strategies in case the agent has no interesting response that is relevant to the context [25].

Table 3: Evaluation on 1-turn and 3-turns dialogue generation using the proposed embedding metrics.

	Twitter			**Ubuntu**		
Model	**Average**	**Greedy**	**Extrema**	**Average**	**Greedy**	**Extrema**
1-turn						
LSTM	0.512	0.389	0.366	0.23	0.169	0.157
HRED	0.501	0.378	0.355	**0.577**	**0.417**	**0.391**
VHRED	**0.533**	**0.396**	**0.38**	0.542	0.384	0.363
3-turns						
LSTM	0.657	0.561	0.374	0.638	0.456	0.378
HRED	0.646	0.552	0.364	0.742	0.524	0.432
VHRED	**0.689**	**0.583**	**0.391**	**0.777**	**0.536**	**0.448**

The above hypotheses are confirmed upon qualitative assessment of the generated responses (table 2). VHRED generates longer and more meaningful responses compared to the LSTM model, which generates mostly generic responses. Additionally, we observed that the VHRED model has learned to better model smilies, slang (see first example in table 2) and can even continue conversations in different languages (see fifth example).[5] Such aspects are not measured by the human study. Further, VHRED appears to be better at generating *stories* or *imaginative actions* compared to the generative baseline models (see third example). The last example in table 2 is a case where the VHRED generated response is more interesting, yet may be less preferred by humans as it is slightly incompatible with the context, compared to the generic LSTM response. In the next section, we back these examples quantitatively, showing that the VHRED model learns to generate *longer* responses with *more information content* that *share semantic similarity* to the context and ground-truth response.

4.3 Results of Metric-based Evaluation

To show the VHRED responses are more on-topic and share semantic similarity to the ground-truth response, we consider three textual similarity metrics based on word embeddings. The *Embedding Average* (Average) metric projects the model response and ground truth response into two separate real-valued vectors by taking the mean over the word embeddings in each response, and then computes the cosine similarity between them [19]. This metric is widely used for measuring textual similarity. The *Embedding Extrema* (Extrema) metric similarly embeds the responses by taking the extremum (maximum of the absolute value) of each dimension, and afterwards computes the cosine similarity between them.The *Embedding Greedy* (Greedy) metric is more fine-grained; it uses cosine similarity between word embeddings to find the closest word in the human-generated response for each word in the model response. Given the (non-exclusive) alignment between words in the two responses, the mean over the cosine similarities is computed for each pair of questions [23]. Since this metric takes into account the alignment between words, it should be more accurate for long responses. While these metrics do not strongly correlate with human judgements of generated responses, we interpret them as measuring *topic similarity*: if the model generated response has similar semantic content to the ground truth human response, then the metrics will yield a high score. To ease reproducibility, we use the publicly available Word2Vec word embeddings trained on the Google News Corpus.[6]

We compute these metrics in two settings: one where the models generate a single response (1-turn), and one where they generate the next three consecutive utterances (3-turns) (table 3). Overall, VHRED seems to better capture the ground truth response topic than either the LSTM or HRED models. The fact that VHRED does better in particular in the setting where the model generates three consecutive utterances strongly suggests that hidden states in both the *decoder* and *context* RNNs of the VHRED models are better able to follow trajectories which remain on-topic w.r.t the dialogue context. This supports our computational hypothesis that the stochastic latent variable helps modulate the training procedure to achieve a better trade-off between short-term and long-term generation. We also observed the same trend when computing the similarity metrics between the model generated responses and the corresponding context, which further reinforces this hypothesis.

[5] There is a notable amount of Spanish and Dutch conversations in the corpus.

[6] `https://code.google.com/archive/p/word2vec/`

Table 4: Response information content on 1-turn generation as measured by average utterance length $|U|$, word entropy $H_w = -\sum_{w \in U} p(w) \log p(w)$ and utterance entropy H_U with respect to the maximum-likelihood unigram distribution of the training corpus p.

	Twitter			**Ubuntu**		
Model	$\lvert U \rvert$	H_w	H_U	$\lvert U \rvert$	H_w	H_U
LSTM	11.21	6.75	75.61	4.27	6.50	27.77
HRED	11.64	6.73	78.35	**11.05**	7.53	**83.16**
VHRED	**12.29**	**6.88**	**84.56**	9.22	**7.70**	71.00
Human	20.57	8.10	166.57	18.30	8.90	162.88

To show that the VHRED responses contain more information content than other model responses, we compute the average response length and average entropy (in bits) w.r.t. the maximum likelihood unigram model over the generated responses (table 4). The unigram entropy is computed on the preprocessed tokenized datasets. VHRED produces responses with higher entropy per word on both Ubuntu and Twitter compared to the HRED and LSTM models. VHRED also produces longer responses overall on Twitter, which translates into responses containing on average 6 bits of information more than the HRED model. Since the actual dialogue responses contain even more information per word than any of the generative models, it reasonable to assume that a higher entropy is desirable. Thus, VHRED compares favourably to recently proposed models in the literature, which often output extremely low-entropy (generic) responses such as *OK* and *I don't know* [24, 15]. Finally, the fact that VHRED produces responses with higher entropy suggests that its responses are on average more diverse than the responses produced by the HRED and LSTM models. This implies that the trajectories of the hidden states of the VHRED model traverse a larger area of the space compared to the hidden states of the HRED and LSTM baselines, which further supports our hypothesis that the stochastic latent variable helps the VHRED model achieve a better trade-off between short-term and long-term generation.

5 Related Work

The use of a stochastic latent variable learned by maximizing a variational lower bound is inspired by the variational autoencoder (VAE) [14, 21]. Such models have been used predominantly for generating images in the continuous domain [11]. However, there has also been recent work applying these architectures for generating sequences, such as the Variational Recurrent Neural Networks (VRNN) [6], which was applied for speech and handwriting synthesis, and Stochastic Recurrent Networks (STORN) [1], which was applied for music generation and motion capture modeling. Both the VRNN and STORN incorporate stochastic latent variables into RNN architectures, but unlike the VHRED they sample a separate latent variable at each time step of the decoder. This does not exploit the hierarchical structure in the data, and thus does not model higher-level variability.

Similar to our work is the Variational Recurrent Autoencoder [7] and the Variational Autoencoder Language Model [3], which apply encoder-decoder architectures to generative music modeling and language modeling respectively. The VHRED model is different from these in the following ways. The VHRED latent variable is conditioned on all previous sub-sequences (sentences). This enables the model to generate multiple sub-sequences (sentences), but it also makes the latent variables co-dependent through the observed tokens. The VHRED model builds on the hierarchical architecture of the HRED model, which makes the model applicable to generation conditioned on long contexts. It has a direct deterministic connection between the *context* and *decoder* RNN, which allows the model to transfer deterministic pieces of information between its components.[7] Crucially, VHRED also demonstrates improved results beyond the autoencoder framework, where the objective is not input reconstruction but the conditional generation of the next utterance in a dialogue.

[7] Our initial experiments confirmed that the deterministic connection between the *context* RNN to the *decoder* RNN was indeed beneficial in terms of lowering the variational bound.

6 Discussion

We have introduced a novel latent variable neural network architecture, called VHRED. The model uses a hierarchical generation process in order to exploit the structure in sequences and is trained using a variational lower bound on the log-likelihood. We have applied the proposed model on the difficult task of dialogue response generation, and have demonstrated that it is an improvement over previous models in several ways, including quality of responses as measured in a human study. The empirical results highlight the advantages of the hierarchical generation process for modelling high-entropy sequences. Finally, it is worth noting that the proposed model is very general. It can in principle be applied to any sequential generation task that exhibits a hierarchical structure, such as document-level machine translation, web query prediction, multi-sentence document summarization, multi-sentence image caption generation, and others.

References

[1] Bayer, J. and Osendorfer, C. (2014). Learning stochastic recurrent networks. In *NIPS Workshop on Advances in Variational Inference*.

[2] Boulanger-Lewandowski, N., Bengio, Y., and Vincent, P. (2012). Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *ICML*.

[3] Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. (2015). Generating sentences from a continuous space. *arXiv:1511.06349*.

[4] Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P., and Robinson, T. (2014). One billion word benchmark for measuring progress in statistical language modeling. In *INTERSPEECH*.

[5] Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *EMNLP*, pages 1724–1734.

[6] Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A., and Bengio, Y. (2015). A recurrent latent variable model for sequential data. In *NIPS*, pages 2962–2970.

[7] Fabius, O. and van Amersfoort, J. R. (2014). Variational recurrent auto-encoders. *arXiv:1412.6581*.

[8] Galley, M., Brockett, C., Sordoni, A., Ji, Y., Auli, M., Quirk, C., Mitchell, M., Gao, J., and Dolan, B. (2015). deltaBLEU: A discriminative metric for generation tasks with intrinsically diverse targets. In *ACL*.

[9] Goodfellow, I., Courville, A., and Bengio, Y. (2015). *Deep Learning*. MIT Press.

[10] Graves, A. (2012). Sequence transduction with recurrent neural networks. In *ICML RLW*.

[11] Gregor, K., Danihelka, I., Graves, A., and Wierstra, D. (2015). DRAW: A recurrent neural network for image generation. In *ICLR*.

[12] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, **9**(8).

[13] Kingma, D. and Ba, J. (2015). Adam: A Method for Stochastic Optimization. In *ICLR*.

[14] Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. In *ICLR*.

[15] Li, J., Galley, M., Brockett, C., Gao, J., and Dolan, B. (2016). A diversity-promoting objective function for neural conversation models. In *NAACL*.

[16] Liu, C.-W., Lowe, R., Serban, I. V., Noseworthy, M., Charlin, L., and Pineau, J. (2016). How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv:1603.08023*.

[17] Lowe, R., Pow, N., Serban, I., and Pineau, J. (2015). The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems. In *SIGDIAL*.

[18] Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048.

[19] Mitchell, J. and Lapata, M. (2008). Vector-based models of semantic composition. In *ACL*, pages 236–244.

[20] Pietquin, O. and Hastie, H. (2013). A survey on metrics for the evaluation of user simulations. *The knowledge engineering review*, **28**(01), 59–73.

[21] Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *ICML*.

[22] Ritter, A., Cherry, C., and Dolan, W. B. (2011). Data-driven response generation in social media. In *EMNLP*, pages 583–593.

[23] Rus, V. and Lintean, M. (2012). A comparison of greedy and optimal assessment of natural language student input using word-to-word similarity metrics. In *Building Educational Applications Workshop, ACL*.

[24] Serban, I. V., Sordoni, A., Bengio, Y., Courville, A. C., and Pineau, J. (2016). Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, pages 3776–3784.

[25] Shaikh, S., Strzalkowski, T., Taylor, S., and Webb, N. (2010). VCA: an experiment with a multiparty virtual chat agent. In *ACL Workshop on Companionable Dialogue Systems*, pages 43–48.

[26] Sordoni, A., Bengio, Y., Vahabi, H., Lioma, C., Simonsen, J. G., and Nie, J.-Y. (2015a). A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *CIKM*.

[27] Sordoni, A., Galley, M., Auli, M., Brockett, C., Ji, Y., Mitchell, M., Nie, J.-Y., Gao, J., and Dolan, B. (2015b). A neural network approach to context-sensitive generation of conversational responses. In *NAACL-HLT*.

[28] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112.

[29] Young, S., Gasic, M., Thomson, B., and Williams, J. D. (2013). POMDP-based statistical spoken dialog systems: A review. *IEEE*, **101**(5), 1160–1179.

Appendix

6.1 Dataset Details

Our Twitter Dialogue Corpus was extracted in 2011. We perform a minimal preprocessing on the dataset to remove irregular punctuation marks and tokenize it using the Moses tokenizer: `https://github.com/moses-smt/mosesdecoder/blob/master/scripts/tokenizer/tokenizer.perl`.

We use the Ubuntu Dialogue Corpus v2.0 extracted in Jamuary 2016 from: `http://cs.mcgill.ca/~jpineau/datasets/ubuntu-corpus-1.0/`. The preprocessed version of the dataset will be made available to the public.

6.2 Model Details

The model implementations will be released to the public upon acceptance of the paper.

Training and Generation

We validate each model on the entire validation set every 5000 training batches.

As mentioned in the main text, at test time we use beam search with 5 beams for outputting responses with the RNN decoders [10]. We define the beam cost as the log-likelihood of the tokens in the beam divided by the number of tokens it contains. This is a well-known modification, which is often applied in machine translation models. In principle, we could sample from the RNN decoders of all the models, but is well known that such sampling produces poor results in comparison to the beam search procedure. It also introduces additional variance into the evaluation procedure, which will make the human study very expensive or even impossible within a limited budget.

Baseline Models

On Ubuntu, the gating function between the *context* RNN and *decoder* RNN is a one-layer feed-forward neural network with hyperbolic tangent activation function.

On Twitter, the HRED *decoder* RNN computes a 1000 dimensional real-valued vector for each hidden time step, which is multiplied with the output *context* RNN. The result is feed through a one-layer feed-forward neural network with hyperbolic tangent activation function, which the *decoder* RNN then takes as input. Furthermore, the *encoder* RNN initial state for each utterance is initialized to the last hidden state of the *encoder* RNN from the previous utterance. We found that this worked slightly better for the VHRED model, but a more careful choice of hyperparameters is likely to make this additional step unnecessary for both the HRED and VHRED models.

Latent Variable Parametrization

We here describe the formal definition of the latent variable prior and approximate posterior distributions. Let $w_1, \ldots, w_T$ be discrete tokens in vocabulary V, which correspond to one sequence (e.g. one dialogue). Let $h_{t,con} \in \mathbb{R}^{d_{h,con}}$ be the hidden state of the HRED *context* encoder at time t. Then the prior mean and covariance matrix are given as:

$$\bar{h}_{t,con} = \text{tanh}(H_{l_2,prior}\text{tanh}(H_{l_1,prior}h_{t,con} + b_{l_1,prior}) + b_{l_2,prior}), \tag{6}$$

$$\boldsymbol{\mu}_{t,\text{prior}} = H_{\mu,prior}\bar{h}_{t,con} + b_{\mu,prior}, \tag{7}$$

$$\Sigma_{t,\text{prior}} = \text{diag}(\log(1 + \exp(H_{\Sigma,prior}\bar{h}_{t,con} + b_{\Sigma,prior}))), \tag{8}$$

where the parameters are $H_{l_1,prior} \in \mathbb{R}^{d_z \times d_{h,con}}$, $H_{\Sigma,prior}, H_{\mu,prior}, H_{l_2,prior} \in \mathbb{R}^{d_z \times d_z}$ and $b_{l_1,prior}, b_{l_2,prior}, b_{\mu,prior}, b_{\Sigma,prior} \in \mathbb{R}^{d_z}$, and where $\text{diag}(\mathbf{x})$ is a function mapping a vector $\mathbf{x}$ to a matrix with diagonal elements $\mathbf{x}$ and all off-diagonal elements equal to zero. At generation time, the latent variable is sampled at the end of each utterance: $\mathbf{z}_t \sim \mathcal{N}(\boldsymbol{\mu}_{t.\text{prior}}, \Sigma_{t,\text{prior}})$.

The equations for the approximate posterior are similar. Let $h_{t,p} \in \mathbb{R}^{d_{h,con}+d_{h,enc}}$ be the concatenation of $h_{t,con}$ and the hidden state of the *encoder* RNN at the end of the next sub-sequence, which we assume has dimensionality $d_{h,enc}$. The approximate posterior mean and covariance matrix are given as:

$$\bar{h}_{t,p} = \text{tanh}(H_{l_2,posterior}\text{tanh}(H_{l_1,posterior}h_{t,p} + b_{l_1,posterior})) + b_{l_2,posterior})), \tag{9}$$

$$\boldsymbol{\mu}_{t,\text{posterior}} = H_{\mu,posterior}\bar{h}_{t,p} + b_{\mu,posterior}, \tag{10}$$

$$\Sigma_{t,\text{posterior}} = \text{diag}(\log(1 + \exp(H_{\Sigma,posterior}\bar{h}_{t,p} + b_{\Sigma,posterior}))), \tag{11}$$

where $H_{l_1,posterior} \in \mathbb{R}^{d_z \times (d_{h,con}+d_{h,enc})}$, $H_{\Sigma,posterior}, H_{\mu,posterior}, H_{l_2,posterior} \in \mathbb{R}^{d_z \times d_z}$ and $b_{l_1,posterior}, b_{l_2,posterior}, b_{\mu,posterior}, b_{\Sigma,posterior} \in \mathbb{R}^{d_z}$ are its parameters. The derivative of the variational bound is computed by sampling one latent variable at the end of each utterance: $\mathbf{z}_t \sim \mathcal{N}(\boldsymbol{\mu}_{t.\text{posterior}}, \Sigma_{t,\text{posterior}})$.

6.3 Model Examples

Model responses for Ubuntu are shown in Table 5. All model responses are available for download at `www.iulianserban.com/Files/UbuntuDialogueCorpus.zip` and `http://www.iulianserban.com/Files/TwitterDialogueCorpus.zip`.

Table 5: Ubuntu model examples. The → token indicates a change of turn.

Context	Response
i set up my hd such that i have to type a passphrase to access it at boot. how can i remove that passwrd, and just boot up normal. i did this at install, it works fine, just tired of having reboots where i need to be at terminal to type passwd in. help? → backup your data, and re-install without encryption " might " be the easiest method	**VHRED:** problem is that i am not able to boot into the live cd, i can access it from the live cd, but i dont know how to use it **HRED:** how would i do that? **LSTM:** how do i do that?
hi, everyone. I download the googleearth. deb, and want to install it with : sudo dpkg -i something. deb, It tells me : google-earth-stable depends on lsb-core (>= 3.2) ; however : Package lsb-core is not installed. So i just install the lsb-core, But it comes error : <url> what should i do? → the error suggests running " sudo apt-get -f install " → i tried that, it comes the same error	**VHRED:** pastebin your /etc/apt/sources. list file **HRED:** try " sudo apt-get -f install " **LSTM:** something
I've a 2.0 TB hard drive. It no longer boots from the hard drive, but I can see it from Gpart and it has a " boot " flag. How do I recover it? → the boot flag doesn't mean anything in Linux why do you not have a backup? you can mount the partition in liveCD and access the data easily (assuming the partition is healthy) → I have one hard drive for the moment (well, I don't count the 250GB drive) besides, I don't have two SATA cables I will make a backup as soon as I get another nice sized hard drive and a SATA cable And how do I find out if the partition is healthy? → boot to ubuntu livecd and try and mount the partition	**VHRED:** fdisk -l gives me a list of partitions **HRED:** How do I do that? **LSTM:** on the live cd?
bottom line is usually if you got 64bit why not use it? (unless there is something that doesn't work on 64bit, which is amitedly not much) → well, now Clip2net is not working... just seems Ubuntu runs more seamlessly on 32-bit and after reading this article, Im wondering if 64 is a bit hyped? <url>	**VHRED:** there is a 64 bit version of ubuntu **HRED:** I have no experience with 64bit, so I don't know what you mean. **LSTM:** something, something, something, something, something
Its comming from my computer 15 people using same router and they have no issues, also it happened recently, before it was ok → yes but your router may have optional settings for dns which if not set to the same dns will override your network interface... it 's an option to consider	**VHRED:** I know, I was just trying to help you. **HRED:** how do I do that? **LSTM:** that is what I was looking for, thanks
it looks as if the network 's DHCP server is dead : " < warn > (eth0) : DHCPv4 request timed out " → but it isn't. windows works	**VHRED:** can you ping the router? **HRED:** that 's what I thought, but I don't know how to fix it **LSTM:**?
I've been having trouble with a new 3TB harddrive and a fresh Ubuntu install. I install (as GPT) and the install looks ok. But then get GRUB " error : out of disk " after reboot. I have an October 2011 PC with updated BIOS and I _do_ see the 3TB. google searches haven't been to helpful so far. has anyone seen GRUB trouble when installing on 3TB harddrives? → have you tried installing grub again and updating it? sudo grub-install sudo update-grub? → I guess I could try that. So I would have to install GRUB from an older Ubuntu install?	**VHRED:** i have no idea, sorry **HRED:** yes, you can install grub to the MBR of your hard drive **LSTM:** grub is installed on the mbr of the hard drive

6.4 Human Study on Amazon Mechanical Turk

Setup

We choose to use crowdsourcing platforms such as AMT rather than carrying out in-lab experiments, even though in-lab experiments usually exhibit less noise and result in higher agreement between human annotators. We do this because AMT experiments involve a larger and more heterogeneous pool of annotators, which implies less cultural and geographic biases, and because such experiments are easier to replicate, which we believe is important for benchmarking future research on these tasks.

Allowing the AMT human evaluators to not assign preference for either response is important, since there are several reasons why humans may not understand the dialogue context, which include topics they are not familiar with, slang language and non-English language. We refer to such evaluations as ‘indeterminable’.

The evaluation setup resembles the classical Turing Test where human judges have to distinguish between human-human conversations and human-computer conversations. However, unlike the original Turing Test, we only ask human evaluators to consider the next utterance in a given conversation and we do not inform them that any responses were generated by a computer. Apart from minimum context and response lengths we impose no restrictions on the generated responses.

Selection Process

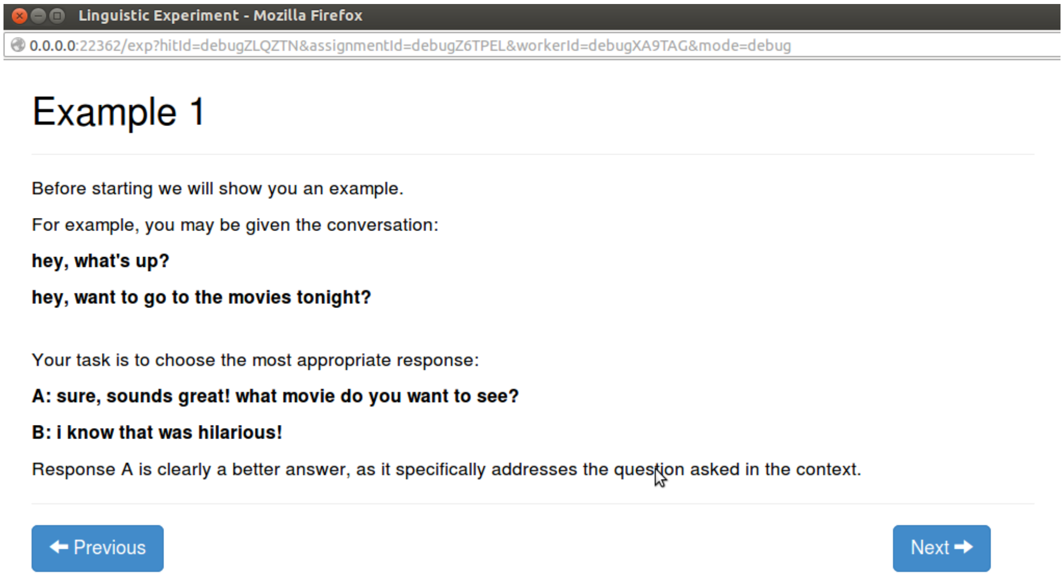
At the beginning of each experiment, we briefly instruct the human evaluator on the task and show them a simple example of a dialogue context and two potential responses. To avoid presentation bias, we shuffle the order of the examples and the order of the potential responses for each example. During each experiment, we also show four trivial ‘attention check’ examples that any human evaluator who has understood the task should be able to answer correctly. We discard responses from human evaluators who fail more than one of these checks.

We select the examples shown to human evaluators at random from the test set. We filter out all non-English conversations and conversations containing offensive content. This is done by automatically filtering out all conversations with non-ascii characters and conversations with profanities, curse words and otherwise offensive content. This filtering is not perfect, so we manually skim through many conversations and filter out conversations with non-English languages and offensive content. On average, we remove about $1/80$ conversations manually. To ensure that the evaluation process is focused on evaluating conditional dialogue response generation (as opposed to unconditional single sentence generation), we constrain the experiment by filtering out examples with fewer than 3 turns in the context. We also filter out examples where either of the two presented responses contain less than 5 tokens. We remove the special token placeholders and apply regex expressions to detokenize the text.

Execution

We run the experiments in batches. For each pairs of models, we carry out $3-5$ human intelligence tests (HITs) on AMT. Each HIT contains $70-90$ examples (dialogue context and two model responses) and is evaluated by $3-4$ unique humans. In total we collect 5363 preferences in 69 HITs.

The following are screenshots from one actual Amazon Mechanical Turk (AMT) experiment. These screenshots show the introduction (debriefing) of the experiment, an example dialogue and one dialogue context with two candidate responses, which human evaluators were asked to choose between. The experiment was carried out using psiturk, which can be downloaded from `www.psiturk.org`. The source code will be released upon publication.


Linguistic Experiment - Mozilla Firefox
0.0.0.0:22362/exp?hitId=debugZLQZTN&assignmentId=debugZ6TPEL&workerId=debugXA9TAG&mode=debug
Previous
Next

Example 1

Before starting we will show you an example.

For example, you may be given the conversation:

hey, what's up?

hey, want to go to the movies tonight?

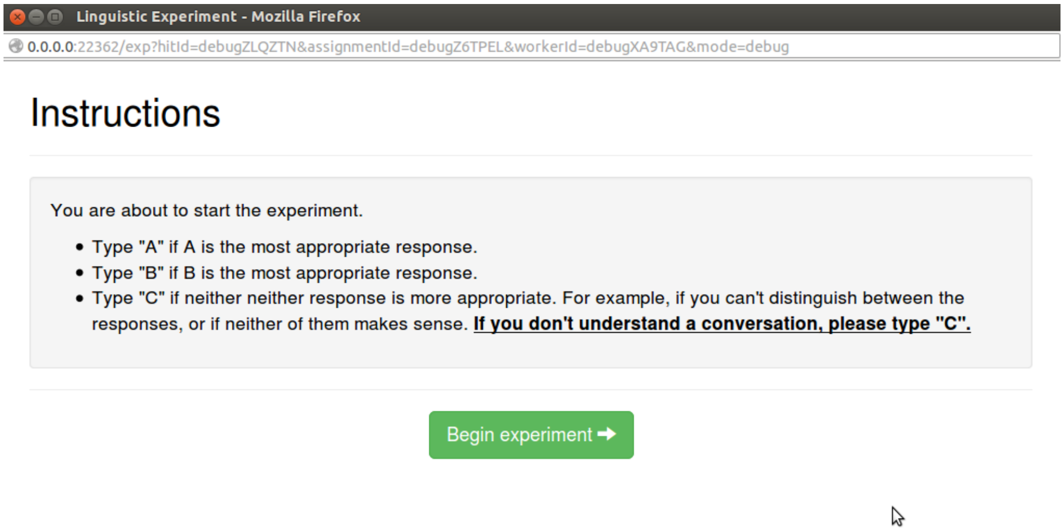
Your task is to choose the most appropriate response:

A: sure, sounds great! what movie do you want to see?

B: i know that was hilarious!

Response A is clearly a better answer, as it specifically addresses the question asked in the context.

Figure 2: Screenshot of the introduction (debriefing) of the experiment.

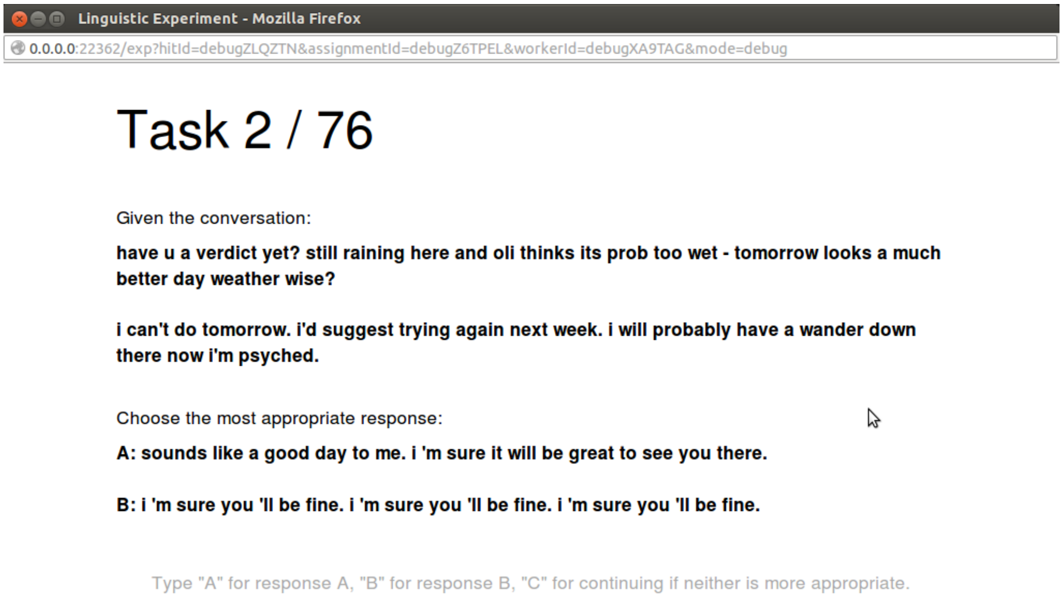

Linguistic Experiment - Mozilla Firefox
0.0.0.0:22362/exp?hitId=debugZLQZTN&assignmentId=debugZ6TPEL&workerId=debugXA9TAG&mode=debug
Begin experiment

Instructions

You are about to start the experiment.

- Type "A" if A is the most appropriate response.
- Type "B" if B is the most appropriate response.
- Type "C" if neither neither response is more appropriate. For example, if you can't distinguish between the responses, or if neither of them makes sense. **If you don't understand a conversation, please type "C".**

Figure 3: Screenshot of the introductory dialogue example.


Linguistic Experiment - Mozilla Firefox
0.0.0.0:22362/exp?hitId=debugZLQZTN&assignmentId=debugZ6TPEL&workerId=debugXA9TAG&mode=debug

Task 2 / 76

Given the conversation:

have u a verdict yet? still raining here and oli thinks its prob too wet - tomorrow looks a much better day weather wise?

i can't do tomorrow. i'd suggest trying again next week. i will probably have a wander down there now i'm psyched.

Choose the most appropriate response:

A: sounds like a good day to me. i 'm sure it will be great to see you there.

B: i 'm sure you 'll be fine. i 'm sure you 'll be fine. i 'm sure you 'll be fine.

Type "A" for response A, "B" for response B, "C" for continuing if neither is more appropriate.

Figure 4: Screenshot of one dialogue context with two candidate responses, which human evaluators were asked to choose between.