Multiresolution Recurrent Neural Networks: An Application to Dialogue Response Generation

Iulian Vlad Serban[*°]
University of Montreal
2920 chemin de la Tour,
Montréal, QC, Canada

Tim Klinger[◇]
IBM Research
T. J. Watson Research Center,
Yorktown Heights,
NY, USA

Gerald Tesauro[◇]
IBM Research
T. J. Watson Research Center,
Yorktown Heights,
NY, USA

Kartik Talamadupula[◇]
IBM Research
T. J. Watson Research Center,
Yorktown Heights,
NY, USA

Bowen Zhou[◇]
IBM Research
T. J. Watson Research Center,
Yorktown Heights,
NY, USA

Yoshua Bengio[†°]
University of Montreal
2920 chemin de la Tour,
Montréal, QC, Canada

Aaron Courville[°]
University of Montreal
2920 chemin de la Tour,
Montréal, QC, Canada

Abstract

We introduce the multiresolution recurrent neural network, which extends the sequence-to-sequence framework to model natural language generation as two parallel discrete stochastic processes: a sequence of high-level coarse tokens, and a sequence of natural language tokens. There are many ways to estimate or learn the high-level coarse tokens, but we argue that a simple extraction procedure is sufficient to capture a wealth of high-level discourse semantics. Such procedure allows training the multiresolution recurrent neural network by maximizing the exact joint log-likelihood over both sequences. In contrast to the standard log-likelihood objective w.r.t. natural language tokens (word perplexity), optimizing the joint log-likelihood biases the model towards modeling high-level abstractions. We apply the proposed model to the task of dialogue response generation in two challenging domains: the Ubuntu technical support domain, and Twitter conversations. On Ubuntu, the model outperforms competing approaches by a substantial margin, achieving state-of-the-art results according to both automatic evaluation metrics and a human evaluation study. On Twitter, the model appears to generate more relevant and on-topic responses according to automatic evaluation metrics. Finally, our experiments demonstrate that the proposed model is more adept at overcoming the sparsity of natural language and is better able to capture long-term structure.

[*] This work was carried out while the first author was at IBM Research.
[°] Email: {iulian.vlad.serban,yoshua.bengio,aaron.courville}@umontreal.ca
[◇] Email: {tklinger,gtesauro,krtalamad,zhou}@us.ibm.com
[†] CIFAR Senior Fellow

1 Introduction

Recurrent neural networks (RNNs) have been gaining popularity in the machine learning community due to their impressive performance on tasks such as machine translation [32, 5] and speech recognition [10]. These results have spurred a cascade of novel neural network architectures [15], including attention [1, 6], memory [34, 9, 15] and pointer-based mechanisms [19].

The majority of the previous work has focused on developing new neural network architectures within the deterministic sequence-to-sequence framework. In other words, it has focused on changing the parametrization of the deterministic function mapping input sequences to output sequences, trained by maximizing the log-likelihood of the observed output sequence. Instead, we pursue a complimentary research direction aimed at generalizing the sequence-to-sequence framework to multiple input and output sequences, where each sequence exhibits its own stochastic process. We propose a new class of RNN models, called multiresolution recurrent neural networks (MrRNNs), which model multiple parallel sequences by factorizing the joint probability over the sequences. In particular, we impose a hierarchical structure on the sequences, such that information from high-level (abstract) sequences flows to low-level sequences (e.g. natural language sequences). This architecture exhibits a new objective function for training: the joint log-likelihood over all observed parallel sequences (as opposed to the log-likelihood over a single sequence), which biases the model towards modeling high-level abstractions. At test time, the model generates first the high-level sequence and afterwards the natural language sequence. This hierarchical generation process enables it to model complex output sequences with long-term dependencies.

Researchers have recently observed critical problems applying end-to-end neural network architectures for dialogue response generation [28, 16]. The neural networks have been unable to generate meaningful responses taking dialogue context into account, which indicates that the models have failed to learn useful high-level abstractions of the dialogue. Motivated by these shortcomings, we apply the proposed model to the task of dialogue response generation in two challenging domains: the goal-oriented Ubuntu technical support domain and non-goal-oriented Twitter conversations. In both domains, the model outperforms competing approaches. In particular, for Ubuntu, the model outperforms competing approaches by a substantial margin according to both a human evaluation study and automatic evaluation metrics achieving a new state-of-the-art result.

2 Model Architecture

2.1 Recurrent Neural Network Language Model

We start by introducing the well-established recurrent neural network language model (RNNLM) [20, 3]. RNNLM variants have been applied to diverse sequential tasks, including dialogue modeling [28], speech synthesis [7], handwriting generation [8] and music composition [4]. Let $w_1, \dots, w_N$ be a sequence of discrete variables, called tokens (e.g. words), such that $w_n \in V$ for vocabulary V. The RNNLM is a probabilistic generative model, with parameters θ, which decomposes the probability over tokens:

$$P_\theta(w_1, \dots, w_N) = \prod_{n=1}^{N} P_\theta(w_n | w_1, \dots, w_{n-1}). \tag{1}$$

where the parametrized approximation of the output distribution uses a softmax RNN:

$$P_\theta(w_{n+1} = v | w_1, \dots, w_n) = \frac{\exp(g(h_n, v))}{\sum_{v' \in V} \exp(g(h_n, v'))}, \tag{2}$$

$$h_n = f(h_{n-1}, w_n),\ g(h_n, v) = O_v^{\mathrm{T}} h_n, \tag{3}$$

where f is the hidden state update function, which we will assume is either the LSTM gating unit [11] or GRU gating unit [5] throughout the rest of the paper. For the LSTM gating unit, we consider the hidden state h_m to be the LSTM cell and cell input hidden states concatenated. The matrix $I \in \mathbb{R}^{d_h \times |V|}$ is the input word embedding matrix, where column j contains the embedding for word index j and $d_h \in \mathbb{N}$ is the word embedding dimensionality. Similarly, the matrix $O \in \mathbb{R}^{d_h \times |V|}$ is the output word embedding matrix. According to the model, the probability of observing a token w at position $n+1$ increases if the context vector h_n has a high dot-product with the word embedding corresponding to token w. Most commonly the model parameters are learned by maximizing the log-likelihood (equivalent to minimizing the cross-entropy) on the training set using gradient descent.

2.2 Hierarchical Recurrent Encoder-Decoder

Our work here builds upon that of Sordoni et al. [31], who proposed the hierarchical recurrent encoder-decoder model (HRED). Their model exploits the hierarchical structure in web queries in order to model a user search session as two hierarchical sequences: a sequence of queries and a sequence of words in each query. Serban et al. [28] continue in the same direction by proposing to exploit the temporal structure inherent in natural language dialogue. Their model decomposes a dialogue into a hierarchical sequence: a sequence of utterances, each of which is a sequence of words. More specifically, the model consists of three RNN modules: an *encoder* RNN, a *context* RNN and a *decoder* RNN. A sequence of tokens (e.g. words in an utterance) are encoded into a real-valued vector by the *encoder* RNN. This in turn is given as input to the *context* RNN, which updates its internal hidden state to reflect all the information up to that point in time. It then produces a real-valued output vector, which the *decoder* RNN conditions on to generate the next sequence of tokens (next utterance). Due to space limitations, we refer the reader to [31, 28] for additional information on the model architecture. The HRED model for modeling structured discrete sequences is appealing for three reasons. First, it naturally captures the hierarchical structure we want to model in the data. Second, the *context* RNN acts like a memory module which can remember things at longer time scales. Third, the structure makes the objective function more stable w.r.t. the model parameters, and helps propagate the training signal for first-order optimization methods [31].

2.3 Multiresolution RNN (MrRNN)

We consider the problem of generatively modeling multiple parallel sequences. Each sequence is hierarchical with the top level corresponding to utterances and the bottom level to tokens. Formally, let $\mathbf{w}_1, \dots, \mathbf{w}_N$ be the first sequence of length N where $\mathbf{w}_n = (w_{n,1}, \dots, w_{n,K_n})$ is the n'th constituent sequence consisting of K_n discrete tokens from vocabulary V^w. Similarly, let $\mathbf{z}_1, \dots, \mathbf{z}_N$ be the second sequence, also of length N, where $\mathbf{z}_n = (z_{n,1}, \dots, z_{n,L_n})$ is the n'th constituent sequence consisting of L_n discrete tokens from vocabulary V^z. In our experiments, each sequence $\mathbf{w}_n$ will consist of the words in a dialogue utterance, and each sequence $\mathbf{z}_n$ will contain the coarse tokens w.r.t. the same utterance (e.g. the nouns in the utterance).

Our aim is to build a probabilistic generative model over all tokens in the constituent sequences $\mathbf{w}_1, \dots, \mathbf{w}_N$ and $\mathbf{z}_1, \dots, \mathbf{z}_N$. Let θ be the parameters of the generative model. We assume that $\mathbf{w}_n$ is independent of $\mathbf{z}_{n'}$ conditioned on $\mathbf{z}_1, \dots, \mathbf{z}_n$ for $n' > n$, and factor the probability over sequences:

$$\begin{aligned} P_\theta(\mathbf{w}_1, \dots, \mathbf{w}_N, \mathbf{z}_1, \dots, \mathbf{z}_N) &= \prod_{n=1}^{N} P_\theta(\mathbf{z}_n | \mathbf{z}_1, \dots, \mathbf{z}_{n-1}) \prod_{n=1}^{N} P_\theta(\mathbf{w}_n | \mathbf{w}_1, \dots, \mathbf{w}_{n-1}, \mathbf{z}_1, \dots, \mathbf{z}_n) \\ &= \prod_{n=1}^{N} P_\theta(\mathbf{z}_n | \mathbf{z}_1, \dots, \mathbf{z}_{n-1}) P_\theta(\mathbf{w}_n | \mathbf{w}_1, \dots, \mathbf{w}_{n-1}, \mathbf{z}_1, \dots, \mathbf{z}_n), \end{aligned} \tag{4}$$

where we define the conditional probabilities over the tokens in each constituent sequence:

$$\begin{aligned} P_\theta(\mathbf{z}_n | \mathbf{z}_1, \dots, \mathbf{z}_{n-1}) &= \prod_{m=1}^{L_n} P_\theta(z_{n,m} | z_{n,1}, \dots, z_{n,m-1}, \mathbf{z}_1, \dots, \mathbf{z}_{n-1}) \\ P_\theta(\mathbf{w}_n | \mathbf{w}_1, \dots, \mathbf{w}_{n-1}, \mathbf{z}_1, \dots, \mathbf{z}_n) &= \prod_{m=1}^{K_n} P_\theta(w_{n,m} | w_{n,1}, \dots, w_{n,m-1}, \mathbf{w}_1, \dots, \mathbf{w}_{n-1}, \mathbf{z}_1, \dots, \mathbf{z}_n) \end{aligned}$$

We refer to the distribution over $\mathbf{z}_1, \dots, \mathbf{z}_N$ as the coarse sub-model, and to the distribution over $\mathbf{w}_1, \dots, \mathbf{w}_N$ as the natural language sub-model. For the coarse sub-model, we parametrize the conditional distribution $P_\theta(\mathbf{z}_n | \mathbf{z}_1, \dots, \mathbf{z}_{n-1})$ as the HRED model described in subsection 2.2, applied to the sequences $\mathbf{z}_1, \dots, \mathbf{z}_N$. For the natural language sub-model, we parametrize $P_\theta(\mathbf{w}_n | \mathbf{w}_1, \dots, \mathbf{w}_{n-1}, \mathbf{z}_1, \dots, \mathbf{z}_n)$ as the HRED model applied to the sequences $\mathbf{w}_1, \dots, \mathbf{w}_N$, but with one difference. The *coarse prediction encoder* GRU-gated RNN encodes all the previously generated tokens $\mathbf{z}_1, \dots, \mathbf{z}_n$ into a real-valued vector, which is concatenated with the *context* RNN

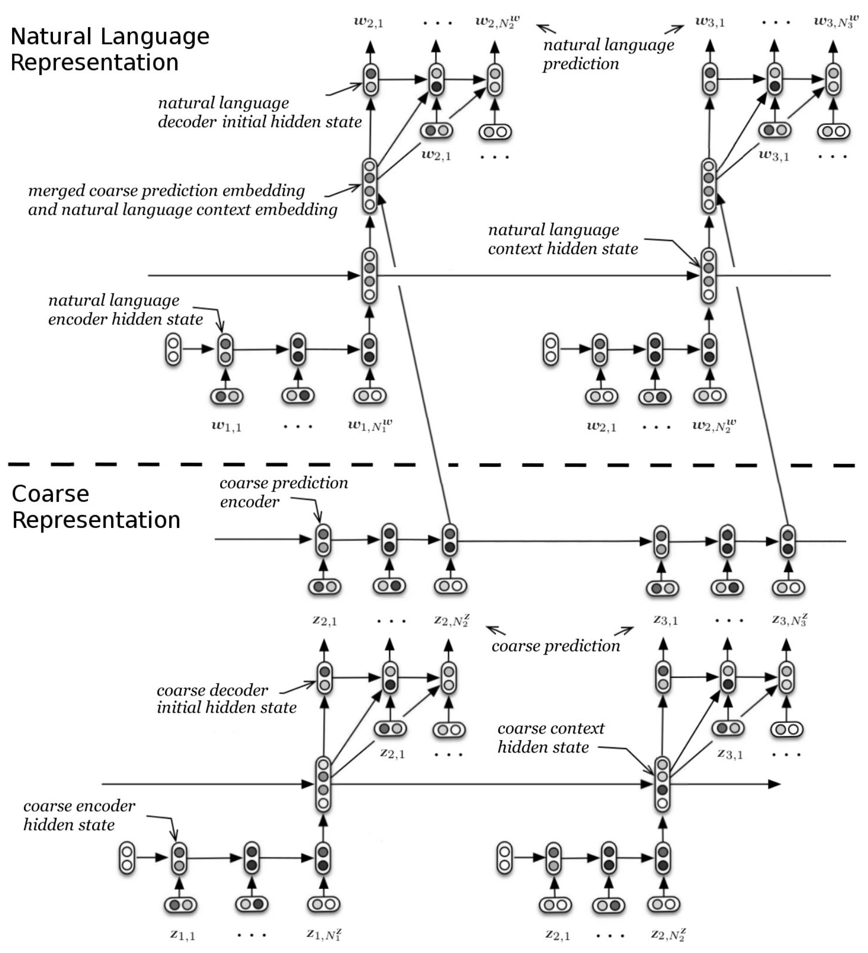
Natural Language Representation
$w_{2,1}$ . . . w_{2,N_2^w}
natural language prediction
$w_{3,1}$ . . . w_{3,N_3^w}
natural language decoder initial hidden state
$w_{2,1}$ . . .
$w_{3,1}$ . . .
merged coarse prediction embedding and natural language context embedding
natural language context hidden state
natural language encoder hidden state
$w_{1,1}$ . . . w_{1,N_1^w}
$w_{2,1}$ . . . w_{2,N_2^w}
Coarse Representation
coarse prediction encoder
$z_{2,1}$ . . . z_{2,N_2^z}
coarse prediction
$z_{3,1}$ . . . z_{3,N_3^z}
coarse decoder initial hidden state
coarse context hidden state
$z_{2,1}$ . . .
$z_{3,1}$ . . .
coarse encoder hidden state
$z_{1,1}$ . . . z_{1,N_1^z}
$z_{2,1}$ . . . z_{2,N_2^z}

Figure 1: Computational graph for the multiresolution recurrent neural network (MrRNN). The lower part models the stochastic process over coarse tokens, and the upper part models the stochastic process over natural language tokens. The rounded boxes represent (deterministic) real-valued vectors, and the variables z and w represent the coarse tokens and natural language tokens respectively.

and given as input to the natural language *decoder* RNN. The *coarse prediction encoder* RNN is important because it encodes the high-level information, which is transmitted to the natural language sub-model. Unlike the *encoder* for the coarse-level sub-model, this encoding will be used to generate natural language and therefore the RNN uses different word embedding parameters. At generation time, the coarse sub-model generates a coarse sequence (e.g. a sequence of nouns), which corresponds to a high-level decision about what the natural language sequence should contain (e.g. nouns to include in the natural language sequence), Conditioned on the coarse sequence, through the *coarse prediction encoder* RNN, the natural language sub-model then generates a natural language sequence (e.g. dialogue utterance). The model is illustrated in Figure 1.

We will assume that both $\mathbf{z}_1, \ldots, \mathbf{z}_N$ and $\mathbf{w}_1, \ldots, \mathbf{w}_N$ are observed and optimize the parameters w.r.t. the joint log-likelihood over both sequences. At test time, to generate a response for sequence n we exploit the probabilistic factorization to approximate the maximum a posteriori (MAP) estimate:

$$\begin{aligned}&\arg\max_{\mathbf{w}_n,\mathbf{z}_n} \; P_\theta(\mathbf{w}_n, \mathbf{z}_n | \mathbf{w}_1, \ldots, \mathbf{w}_{n-1}, \mathbf{z}_1, \ldots, \mathbf{z}_{n-1}) \\ &\approx \arg\max_{\mathbf{w}_n} P_\theta(\mathbf{w}_n | \mathbf{w}_1, \ldots, \mathbf{w}_{n-1}, \mathbf{z}_1, \ldots, \mathbf{z}_{n-1}, \mathbf{z}_n) \arg\max_{\mathbf{z}_n} P_\theta(\mathbf{z}_n | \mathbf{z}_1, \ldots, \mathbf{z}_{n-1}), \qquad (5)\end{aligned}$$

where we further approximate the MAP for each constituent sequence using beam search.

3 Tasks

We consider the task of natural language response generation for dialogue. Dialogue systems have been developed for applications ranging from technical support to language learning and entertainment [35, 30]. Dialogue systems can be categorized into two different types: goal-driven dialogue systems and non-goal-driven dialogue systems [27]. To demonstrate the versatility of the MrRNN, we apply it to both goal-driven and non-goal-driven dialogue tasks. We focus on the task of conditional response generation. Given a dialogue context consisting of one or more utterances, the model must generate the next response in the dialogue.

Ubuntu Dialogue Corpus The goal-driven dialogue task we consider is technical support for the Ubuntu operating system, where we use the Ubuntu Dialogue Corpus [18]. The corpus consists of about 0.5 million natural language dialogues extracted from the *#Ubuntu* Internet Relayed Chat (IRC) channel. Users entering the chat channel usually have a specific technical problem. The users first describe their problem and afterwards other users try to help them resolve it. The technical problems range from software-related issues (e.g. installing or upgrading existing software) and hardware-related issues (e.g. fixing broken drivers or partitioning hard drives) to informational needs (e.g. finding software with specific functionality). Additional details are given in appendix 8.

Twitter Dialogue Corpus The next task we consider is the non-goal-driven task of generating responses to Twitter conversations. We use a Twitter dialogue corpus extracted in the first half of 2011 using a procedure similar to Ritter et al. [24]. Unlike the Ubuntu domain, Twitter conversations are often more noisy and do not necessarily center around a single topic. We perform a minimal preprocessing on the dataset to remove irregular punctuation marks and afterwards tokenize it. The dataset is split into training, validation and test sets containing respectively $749,060$, $93,633$ and $10,000$ dialogues.

4 Coarse Sequence Representations

We experiment with two procedures for extracting the coarse sequence representations:

Noun Representation This procedure aims to exploit the basic high-level structure of natural language discourse.It is based on the hypothesis that dialogues are topic-driven and that these topics may be characterized by nouns. In addition to a tokenizer, used by both the HRED and RNNLM model, it requires a part-of-speech (POS) tagger to identify the nouns in the dialogue. The procedure uses a set of 84 and 795 predefined stop words for Ubuntu and Twitter respectively. It maps a natural language utterance to its coarse representation by extracting all the nouns using the POS tagger and then removing all stop words and repeated words (keeping only the first occurrence of a word). Dialogue utterances without nouns are assigned the "no_nouns" token. The procedure also extracts the tense of each utterance and adds it to the beginning of the coarse representation.

Activity-Entity Representation This procedure is specific to the Ubuntu technical support task, for which it aims to exploit domain knowledge related to technical problem solving. It is motivated by the observation that most dialogues are centered around *activities* and *entities*. For example, it is very common for users to state a specific problem they want to resolve, e.g. *how do I install program X?* or *My driver X doesn't work, how do I fix it?* In response to such questions, other users often respond with specific instructions, e.g. *Go to website X to download software Y* or *Try to execute command X*. In such cases, it is clear that the principal information resides in the technical entities and in the verbs (e.g. *install*, *fix*, *download*), and therefore that it will be advantageous to explicitly model this structure. Motivated by this observation, the procedure uses a set of 192 activities (verbs), created by manual inspection, and a set of 3115 technical entities and 230 frequent terminal commands, extracted automatically from available package managers and from the web. The procedure uses the POS tagger to extract the verbs from the each natural language utterance. It maps the natural language to its coarse representation by keeping only verbs from the activity set, as well as entities from the technical entity set (irrespective of their POS tags). If no activity is found in an utterance, the representation is assigned the "none_activity" token. The procedure also appends a binary variable to the end of the coarse representation indicating if a terminal command was detected in the utterance. Finally, the procedure extracts the tense of each utterance and adds it to the beginning of the coarse representation.

Both extraction procedures are applied at the utterance level, therefore there exists a one-to-one alignment between coarse sequences and natural language sequences (utterances). There also exists a one-to-many alignment between the coarse sequence tokens and the corresponding natural language tokens, with the exception of a few special tokens. Further details are given in appendix 9.

5 Experiments

We optimize all models based on the training set joint log-likelihood over coarse sequences and natural language sequences using the first-order stochastic gradient optimization method Adam [13]. We train all models using early stopping with patience on the joint-log-likelihood [2]. We choose our hyperparameters based on the joint log-likelihood of the validation set. We define the $20K$ most frequent words as the vocabulary and the word embedding dimensionality to size 300 for all models, with the exception of the RNNLM and HRED on Twitter, where we use embedding dimensionality of size 400. We apply gradient clipping to stop the parameters from exploding [23]. At test time, we use a beam search of size 5 for generating the model responses. Further details are given in appendix 10

5.1 Baseline Models

We compare our models to several baselines used previously in the literature. The first is the standard RNNLM with LSTM gating function [20] (LSTM), which at test time is similar to the Seq2Seq LSTM model [32]. The second baseline is the HRED model with LSTM gating function for the *decoder* RNN and GRU gating function for the *encoder* RNN and *context* RNN, proposed for dialogue response generation by Serban et al.[28] [31]. Source code for both baseline models will be made publicly available upon acceptance for publication. For both Ubuntu and Twitter, we specify the RNNLM model to have 2000 hidden units with the LSTM gating function. For Ubuntu, we specify the HRED model to have 500, 1000 and 500 hidden units respectively for the *encoder* RNN, *context* RNN and *decoder* RNN. For Twitter, we specify the HRED model to have 2000, 1000 and 1000 hidden units respectively for the *encoder* RNN, *context* RNN and *decoder* RNN. The third baseline is the latent variable latent variable hierarchical recurrent encoder-decoder (VHRED) proposed by Serban et al. [29]. We use the exact same VHRED models as Serban et al. [29].

For Ubuntu, we introduce a fourth baseline, called *HRED + Activity-Entity Features*, which has access to the past activity-entity pairs. This model is similar to to the natural language sub-model of the MrRNN model, with the difference that the natural language *decoder* RNN is conditioned on a real-valued vector, produced by a GRU RNN encoding *only* the past coarse-level activity-entity sub-sequences. This baseline helps differentiate between a model which observes the coarse-level sequences only as as additional features and a model which explicitly models the stochastic process of the coarse-level sequences. We specify the model to have 500, 1000, 2000 hidden units respectively for the *encoder* RNN, *context* RNN and *decoder* RNN. We specify the GRU RNN encoding the past coarse-level activity-entity sub-sequences to have 500 hidden units.

5.2 Multiresolution RNN

The coarse sub-model is parametrized as the Bidirectional-HRED model [28] with 1000, 1000 and 2000 hidden units respectively for the coarse-level *encoder*, *context* and *decoder* RNNs. The natural language sub-model is parametrized as a conditional HRED model with 500, 1000 and 2000 hidden units respectively for the natural language *encoder*, *context* and *decoder* RNNs. The *coarse prediction encoder* RNN GRU RNN is parametrized with 500 hidden units.

5.3 Ubuntu

Evaluation Methods It has long been known that accurate evaluation of dialogue system responses is difficult [26]. Liu et al. [17] have recently shown that all automatic evaluation metrics adapted for such evaluation, including word overlap-based metrics such as BLEU and METEOR, have either very low or no correlation with human judgment of the system performance. We therefore carry out an in-lab human study to evaluate the Ubuntu models. We recruit 5 human evaluators, and show them

Table 1: Ubuntu evaluation using precision (P), recall (R), F1 and accuracy metrics w.r.t. activity, entity, tense and command (Cmd) on ground truth utterances, and human fluency and relevancy scores given on a scale 0-4 (* indicates scores significantly different from baseline models at 90% confidence)

	Activity			**Entity**			**Tense**	**Cmd**	**Human Eval.**	
Model	**P**	**R**	**F1**	**P**	**R**	**F1**	**Acc.**	**Acc.**	**Fluency**	**Relevancy**
LSTM	1.7	1.03	1.18	1.18	0.81	0.87	14.57	94.79	-	-
HRED	5.93	4.05	4.34	2.81	2.16	2.22	22.2	92.58	2.98	1.01
VHRED	6.43	4.31	4.63	3.28	2.41	2.53	20.2	92.02	-	-
HRED + Act.-Ent.	7.15	5.5	5.46	3.03	2.43	2.44	28.02	86.69	2.96	0.75
MrRNN Noun	5.81	3.56	4.04	**8.68**	**5.55**	**6.31**	24.03	90.66	**3.48***	**1.32***
MrRNN Act.-Ent.	**16.84**	**9.72**	**11.43**	4.91	3.36	3.72	**29.01**	**95.04**	**3.42***	1.04

each $30 - 40$ dialogue contexts with the ground truth response and 4 candidate responses (HRED, HRED + Activity-Entity Features and MrRNNs). For each context example, we ask them to compare the candidate responses to the ground truth response and dialogue context, and rate them for fluency and relevancy on a scale $0 - 4$. Our setup is very similar to the evaluation setup used by Koehn and Monz [14], and comparable to Liu et al [17]. Further details are given in appendix 11.

We further propose a new set of metrics for evaluating model responses on Ubuntu, which compare the activities and entities in the model generated response with those of the ground truth response. That is, the ground truth and model responses are mapped to their respective activity-entity representations, using the automatic procedure discussed in section 4, and then the overlap between their activities and entities are measured according to precision, recall and F1-score. Based on a careful manual inspection of the extracted activities and entities, we believe that these metrics are particularly suited for the goal-oriented Ubuntu Dialogue Corpus. The activities and entities reflect the principal instructions given in the responses, which are key to resolving the technical problems. Therefore, a model able to generate responses with actions and entities similar to the ground truth human responses – which often do lead to solving the users problem – is more likely to yield a successful dialogue system. The reader is encouraged to verify the details and completeness of the activity-entity representations in appendix 9.

Results The results on Ubuntu are given in table 1. The MrRNNs clearly perform substantially better than the baseline models both w.r.t. human evaluation and automatic evaluation metrics. The MrRNN with noun representations achieves $2x - 3x$ higher scores w.r.t. entities compared to other models, and the human evaluators also rate its fluency and relevancy substantially higher than other models. The MrRNN with activity representations achieves $2x - 3x$ higher scores w.r.t. activities compared to other models and nearly $2x$ higher scores w.r.t. entities compared to all baselines. Human evaluators also rate its fluency substantially higher than the baseline models. However,its relevancy is rated only slightly higher compared to the HRED model, which we believe is caused by human evaluators being more likely to noticing software entities than actions in the dialogue responses (even though actions are critical to solving the actual technical problem). Overall, the results demonstrate that the MrRNNs have learned to model high-level goal-oriented sequential structure on Ubuntu.

Model responses are shown in Table 2. In general, the MrRNN responses are more coherent and topic-oriented compared to the LSTM and HRED responses, which usually produce very generic responses [28]. In particular, the MrRNN with activity-entity representation appears to give more goal-oriented instructions compared to the MrRNN with noun representation (see examples 2-4 in Table 2). Additional examples are shown in appendix 12.

Table 2: Ubuntu model examples. The arrows indicate change of turn.

Context	**Response**
Hey guys what do you in general use for irc something ubuntu xchat or xchat-gnome ? → without -gnome. that is just cut down → you mean drop xchat-gnome and go with xchat ?	**MrRNN Act. -Ent.:** im using xchat right now **MrRNN Noun:** what is xchat-gnome ? **VHRED:** correct **HRED:** yes
when setting up rules with iptables command only writes changes this file " etciptables. rules "? i ask this so i can backup before messing anything → sudo iptables-save something . dat to backup your rules restore with sudo iptables-restore < something . dat	**MrRNN Act. -Ent.:** I don't see any reason why you need iptables to do that **MrRNN Noun:** are you using ubuntu ? **VHRED:** thx **HRED:** thanks
Hello I tried to install ubuntu studio but I get W : Failed to fetch <url >404 Not Found when apt-get update → the ppa you added is not online	**MrRNN Act. -Ent.:** try a different mirror **MrRNN Noun:** something **VHRED:** but I dont know how to add it **HRED:** how do I do that ?
hi community.. is there any difference between " /.bashrc " " /.bashrc. local " → yes, they're different files :-) the default /.bashrc sources /.bashrc. local → sorry I could undersatnd... If I write on terminal " gedit /.bashrc " it opens a file but how I open " /.bashrc. local " → " gedit /.bashrc. local "... → but it just open blank file nothing inside it	**MrRNN Act. -Ent.:** open up gedit /.bashrc, there is a file called " something " **MrRNN Noun:** the file is empty.. **VHRED:** it's in /.bashrc or /.bashrc **HRED:** what are you trying to do ?

Table 3: Twitter evaluation using embedding metrics (mean scores $\pm$ 95% confidence intervals)

Model	**Average**	**Greedy**	**Extrema**
LSTM	51.24 ± 0.51	38.9 ± 0.39	36.58 ± 0.36
HRED	50.1 ± 0.52	37.83 ± 0.4	35.55 ± 0.37
VHRED	53.26 ± 0.45	39.64 ± 0.34	37.98 ± 0.32
MrRNN	$\mathbf{68.62 \pm 0.22}$	$\mathbf{56 \pm 0.17}$	$\mathbf{43.06 \pm 0.2}$

5.4 Twitter

Evaluation Methods For Twitter, we use the three embedding-based textual similarity metrics proposed by Liu et al. [17]: *Embedding Average* (Average), *Embedding Extrema* (Extrema) and *Embedding Greedy* (Greedy). All three metrics are based on computing the textual similarity between the ground truth response and the model response using on word embeddings. All three metrics measure topic similarity: if a model-generated response is on the same topic as the ground truth response (e.g. contain paraphrases of the same words), the metrics will yield a high score. This is a highly desirable property for dialogue systems on an open platform such as Twitter, however it is also substantially different from measuring the overall dialogue system performance, or the appropriateness of a single response, which would require human evaluation.

Results The results on Twitter are given in table 3. The responses of the MrRNN with noun representation are better than all other models. In accordance with our previous results, this indicates that the model has learned to generate more on-topic responses and, thus, that explicitly modeling the stochastic process over nouns helps learn the high-level structure.

6 Related Work

Closely related to our work is the model proposed by Ji et al.[12], which jointly models natural language text and high-level discourse phenomena. However, it only models a discrete class per sentence at the high level, which must be manually annotated by humans. On the other hand, MrRNN models a sequence of automatically extracted high-level tokens. Recurrent neural network models with stochastic latent variables, such as the Variational Recurrent Neural Networks by Chung et al. [7], are also closely related to our work. These models face the more difficult task of learning the high-level representations, while simultaneously learning to model the generative process over high-level sequences and low-level sequences, which is a more difficult optimization problem. In addition to this, such models assume the high-level latent variables to be continuous, usually Gaussian, distributions.

Recent dialogue-specific neural network architectures, such as the model proposed by Wen et al. [33], are also relevant to our work. Different from the MrRNN, they require domain-specific hand-crafted high-level (dialogue state) representations with human-labelled examples, and they usually consist of several sub-components each trained with a different objective function.

7 Discussion

We have proposed the multiresolution recurrent neural network (MrRNN) for generatively modeling sequential data at multiple levels of abstraction. It is trained by optimizing the joint log-likelihood over the sequences at each level. We apply MrRNN to dialog response generation on two different tasks, Ubuntu technical support and Twitter conversations, and evaluate it in a human evaluation study and via automatic evaluation metrics. On Ubuntu, MrRNN demonstrates dramatic improvements compared to competing models. On Twitter, MrRNN appears to generate more relevant and on-topic responses. Even though abstract information is implicitly present in natural language dialogues, by explicitly representing information at different levels of abstraction and jointly optimizing the generation process across abstraction levels, MrRNN is able to generate more fluent, relevant and goal-oriented responses. The results suggest that the fine-grained abstraction (low-level) provides the architecture with increased fluency for predicting natural utterances, while the coarse-grained (high-level) abstraction gives it the semantic structure necessary to generate more coherent and relevant utterances. The results also imply that it is not simply a matter of adding additional features for prediction – MrRNN outperforms a competitive baseline augmented with the coarse-grained abstraction sequences as features – rather, it is the combination of representation and generation at multiple levels that yields the improvements. Finally, we observe that the architecture provides a general framework for modeling discrete sequences, as long as a coarse abstraction is available. We therefore conjecture that the architecture may successfully be applied to broader natural language generation tasks, such as generating prose and persuasive argumentation, and other tasks involving discrete sequences, such as music composition. We leave this to future work.

References

[1] Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *ICLR*.

[2] Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. In *Neural Networks: Tricks of the Trade*, pages 437–478. Springer.

[3] Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *The Journal of Machine Learning Research*, **3**, 1137–1155.

[4] 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*.

[5] Cho, K. *et al.* (2014). Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proc. of EMNLP*, pages 1724–1734.

[6] Chorowski, J. K., Bahdanau, D., Serdyuk, D., Cho, K., and Bengio, Y. (2015). Attention-based models for speech recognition. In *NIPS*, pages 577–585.

[7] 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.

[8] Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv:1308.0850*.

[9] Graves, A., Wayne, G., and Danihelka, I. (2014). Neural turing machines. *arXiv:1410.5401*.

[10] Hinton, G. *et al.* (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, **29**(6), 82–97.

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

[12] Ji, Y., Haffari, G., and Eisenstein, J. (2016). A latent variable recurrent neural network for discourse relation language models. In *NAACL-HLT*.

[13] Kingma, D. and Ba, J. (2015). Adam: A method for stochastic optimization. In *Proc. of ICLR*.

[14] Koehn, P. and Monz, C. (2006). Manual and automatic evaluation of machine translation between european languages. In *Workshop on Statistical Machine Translation, ACL*, pages 102–121.

[15] Kumar, A., Irsoy, O., Su, J., Bradbury, J., English, R., Pierce, B., Ondruska, P., Gulrajani, I., and Socher, R. (2016). Ask me anything: Dynamic memory networks for natural language processing. *ICML*.

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

[17] 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*.

[18] 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 *Proc. of SIGDIAL-2015*.

[19] Luong, M. T., Sutskever, I., Le, Q. V., Vinyals, O., and Zaremba, W. (2015). Addressing the rare word problem in neural machine translation. In *ACL*.

[20] Mikolov, T. *et al.* (2010). Recurrent neural network based language model. In *11th Proceedings of INTERSPEECH*, pages 1045–1048.

[21] Mikolov, T. *et al.* (2013). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

[22] Owoputi, O. *et al.* (2013). Improved part-of-speech tagging for online conversational text with word clusters. In *Proc. of ACL*.

[23] Pascanu, R., Mikolov, T., and Bengio, Y. (2012). On the difficulty of training recurrent neural networks. *Proceedings of the 30th International Conference on Machine Learning*, **28**.

[24] Ritter, A., Cherry, C., and Dolan, W. B. (2011a). Data-driven response generation in social media. In *EMNLP*.

[25] Ritter, A., Clark, S., Mausam, and Etzioni, O. (2011b). Named entity recognition in tweets: An experimental study. In *Proc. of EMNLP*, pages 1524–1534.

[26] Schatzmann, J., Georgila, K., and Young, S. (2005). Quantitative evaluation of user simulation techniques for spoken dialogue systems. In *6th SIGdial Workshop on DISCOURSE and DIALOGUE*.

[27] Serban, I. V., Lowe, R., Charlin, L., and Pineau, J. (2015). A survey of available corpora for building data-driven dialogue systems. *CoRR*, **abs/1512.05742**.

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

[29] Serban, I. V., Sordoni, A., Lowe, R., Charlin, L., Pineau, J., Courville, A., and Bengio, Y. (2016b). A hierarchical latent variable encoder-decoder model for generating dialogues. *arXiv preprint arXiv:1605.06069*.

[30] Shawar, B. A. and Atwell, E. (2007). Chatbots: are they really useful? In *LDV Forum*, volume 22.

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

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

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

[34] Weston, J., Chopra, S., and Bordes, A. (2015). Memory networks. *ICLR*.

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

Appendix

8 Task Details

Ubuntu We use the Ubuntu Dialogue Corpus v2.0 extracted Jamuary, 2016: `http://cs.mcgill.ca/~jpineau/datasets/ubuntu-corpus-1.0/`.

Twitter We preprocess the dataset using the Moses tokenizer extracted June, 2015: `https://github.com/moses-smt/mosesdecoder/blob/master/scripts/tokenizer/tokenizer.perl`.[1]

9 Coarse Sequence Representations

Nouns

The noun-based procedure for extracting coarse tokens aims to exploit high-level structure of natural language discourse. More specifically, it builds on the hypothesis that dialogues in general are topic-driven and that these topics may be characterized by the nouns inside the dialogues. At any point in time, the dialogue is centered around one or several topics. As the dialogue progresses, the underlying topic evolves as well. In addition to the tokenizer required by the previous extraction procedure, this procedure also requires a part-of-speech (POS) tagger to identify the nouns in the dialogue suitable for the language domain.

For extracting the noun-based coarse tokens, we define a set of 795 stop words for Twitter and 84 stop words for Ubuntu containing mainly English pronouns, punctuation marks and prepositions (excluding special placeholder tokens). We then extract the coarse tokens by applying the following procedure to each dialogue:

1. We apply the POS tagger version 0.3.2 developed by Owoputi and colleagues [22] to extract POS.[2] For Twitter, we use the parser trained on the Twitter corpus developed by Ritter et al. [25]. For Ubuntu, we use the parser trained on the NPS Chat Corpus developed by Forsyth and Martellwhich was extracted from IRC chat channels similar to the Ubuntu Dialogue Corpus.[3][4]
2. Given the POS tags, we remove all words which are not tagged as nouns and all words containing non-alphabet characters.[5]. We keep all urls and paths.
3. We remove all stop words and all repeated tokens, while maintaining the order of the tokens.
4. We add the "no_nouns" token to all utterances, which do not contain any nouns. This ensures that no coarse sequences are empty. It also forces the coarse sub-model to explicitly generate at least one token, even when there are no actual nouns to generate.
5. For each utterance, we use the POS tags to detect three types of time tenses: past, present and future tenses. We append a token indicating which of the 2^3 tenses are present at the beginning of each utterance.[6] If no tenses are detected, we append the token "no_tenses".

As before, there exists a one-to-many alignment between the extracted coarse sequence tokens and the natural language tokens, since this procedure also maintains the ordering of all special placeholder tokens, with the exception of the "no_nouns" token.

We cut-off the vocabulary at 10000 coarse tokens for both the Twitter and Ubuntu datasets excluding the special placeholder tokens. On average a Twitter dialogue in the training set contains 25 coarse tokens, while a Ubuntu dialogue in the training set contains 36 coarse tokens.

[1]Due to Twitter's Terms and Conditions we are unfortunately not allowed to publish the preprocessed dataset.

[2]`www.cs.cmu.edu/~ark/TweetNLP/`

[3]As input to the POS tagger, we replace all unknown tokens with the word "something" and remove all special placeholder tokens (since the POS tagger was trained on a corpus without these words). We further reduce any consecutive sequence of spaces to a single space. For Ubuntu, we also replace all commands and entities with the word "something". For Twitter, we also replace all numbers with the word "some", all urls with the word "somewhere" and all heart *emoticons* with the word "love".

[4]Forsyth, E. N. and Martell, C. H. (2007). Lexical and discourse analysis of online chat dialog. In Semantic Computing, 2007. ICSC 2007. International Conference on, pages 19–26. IEEE.

[5]We define nouns as all words with tags containing the prefix "NN" according to the PTB-style tagset.

[6]Note that an utterance may contain several sentences. It therefore often happens that an utterance contains several time tenses.

Activity-Entity Pairs

The activity-entity-based procedure for extracting coarse tokens attempts to exploit domain specific knowledge for the Ubuntu Dialogue Corpus, in particular in relation to providing technical assistance with problem solving. Our manual inspection of the corpus shows that many dialogues are centered around *activities*. For example, it is very common for users to state a specific problem they want to resolve, e.g. *how do I install program X?* or *My driver X doesn't work, how do I fix it?*. In response to such queries, other users often respond with specific instructions, e.g. *Go to website X to download software Y* or *Try to execute command X*. In addition to the technical entities, the principle message conveyed by each utterance resides in the verbs, e.g. *install*, *work*, *fix*, *go*, *to*, *download*, *execute*. Therefore, it seems clear that a dialogue system must have a strong understanding of both the activities and technical entities if it is to effectively assist users with technical problem solving. It seems likely that this would require a dialogue system able to relate technical entities to each other, e.g. to understand that *firefox* depends on the *GCC* library, and conform to the temporal structure of activities, e.g. understanding that the *install* activity is often followed by *download* activity.

We therefore construct two word lists: one for activities and one for technical entities. We construct the activity list based on manual inspection yielding a list of 192 verbs. For each activity, we further develop a list of synonyms and conjugations of the tenses of all words. We also use Word2Vec word embeddings [21], trained on the Ubuntu Dialogue Corpous training set, to identify commonly misspelled variants of each activity. The result is a dictionary, which maps a verb to its corresponding activity (if such exists). For constructing the technical entity list, we scrape publicly available resources, including Ubuntu and Linux-related websites as well as the Debian package manager *APT*. Similar to the activities, we also use the Word2Vec word embeddings to identify misspelled and paraphrased entities. This results in another dictionary, which maps one or two words to the corresponding technical entity. In total there are 3115 technical entities. In addition to this we also compile a list of 230 frequent commands. Examples of the extracted activities, entities and commands can be found in the appendix.

Afterwards, we extract the coarse tokens by applying the following procedure to each dialogue:

1. We apply the technical entity dictionary to extract all technical entities.
2. We apply the POS tagger version 0.3.2 developed by Owoputi and colleagues, trained on the NPS Chat Corpus developed by Forsyth and Martell as before. As input to the POS tagger, we map all technical entities to the token "something". This transformation should improve the POS tagging accuracy, since The corpus the parser was trained on does not contain technical words.
3. Given the POS tags, we extract all verbs which correspond to activities.[7]. If there are no verbs in an entire utterance and the POS tagger identified the first word as a noun, we will assume that the first word is in fact a verb. We do this, because the parser does not work well for tagging technical instructions in imperative form, e.g. *upgrade firefox*. If no activities are detected, we append the token "none_activity" to the coarse sequence. We also keep all urls and paths.
4. We remove all repeated activities and technical entities, while maintaining the order of the tokens.
5. If a command is found inside an utterance, we append the "cmd" token at the end of the utterance. Otherwise, we append the "no_cmd" token to the end of the utterance. This enables the coarse sub-model to predict whether or not an utterance contains executable commands.
6. As for the noun-based coarse representation, we also append the time tense to the beginning of the sequence.

As before, there exists a one-to-many alignment between the extracted coarse sequence tokens and the natural language tokens, with the exception of the "none_activity" and "no_cmd" tokens.

Since the number of unique tokens are smaller than 10000, we do not need to cut-off the vocabulary. On average a Ubuntu dialogue in the training set contains 43 coarse tokens.

Our manual inspection of the extracted coarse sequences, show that the technical entities are identified with very high accuracy and that the activities capture the main intended action in the majority of utterances. Due to the high quality of the extracted activities and entities, we are confident that they may be used for evaluation purposes as well.

[7]We define verbs as all words with tags containing the prefix "VB" according to the PTB-style tagset.

Table 4: Twitter Coarse Sequence Examples

Natural Language Tweets	Noun Representation
<first_speaker> at pinkberry with my pink princess enjoying a precious moment <url> <second_speaker>- they are adorable , alma still speaks about emma bif sis . hugs	present_tenses pinkberry princess moment present_tenses alma emma bif sis hugs
<first_speaker> <at> when you are spray painting , where are you doing it ? outside ? in your apartment ? where ? <second_speaker> <at> mostly spray painting outside but some little stuff in the bathroom .	present_tenses spray painting apartment present_tenses spray stuff bathroom

Table 5: Ubuntu Coarse Sequence Examples

Natural Language Dialogues	Activity-Entity Coarse Dialogues
if you can get a hold of the logs , there 's stuff from **unknown** about his inability to install amd64 I'll check fabbione 's log , thanks sounds like he had the same problem I did ew , why ? ... upgrade lsb-base and acpid i'm up to date what error do you get ? i don't find error :/ where do i search from ? acpid works , but i must launch it manually in a root sterm ...	future_tenses get_activity install_activity amd64_entity no_cmd no_tenses check_activity no_cmd past_present_tenses none_activity no_cmd no_tenses none_activity no_cmd ... no_tenses upgrade_activity lsb_entity acpid_entity no_cmd no_tenses none_activity no_cmd present_tenses get_activity no_cmd present_tenses discover_activity no_cmd present_future_tenses work_activity acpid_entity root_entity no_cmd ...

Stop Words for Noun-based Coarse Tokens

Ubuntu stop words for noun-based coarse representation:

all another any anybody anyone anything both each each other either everybody everyone everything few he her hers herself him himself his I it its itself many me mine more most much myself neither no one nobody none nothing one one another other others ours ourselves several she some somebody someone something that their theirs them themselves these they this those us we what whatever which whichever who whoever whom whomever whose you your yours yourself yourselves . , ? ' - – !

Twitter stop words for noun-based coarse representation: [8]

all another any anybody anyone anything both each each other either everybody everyone everything few he her hers herself him himself his I it its itself many me mine more most much myself neither no one nobody none nothing one one another other others ours ourselves several she some somebody someone something that their theirs them themselves these they this those us we what whatever which whichever who whoever whom whomever whose you your yours yourself yourselves . , ? ’ - – !able about above abst accordance according accordingly across act actually added adj adopted affected affecting affects after afterwards again against ah all almost alone along already also although always am among amongst an and announce another any anybody anyhow anymore anyone anything anyway anyways anywhere apparently approximately are aren arent arise around as aside ask asking at auth available away awfully b back bc be became because become becomes becoming been before beforehand begin beginning beginnings begins behind being believe below beside besides between beyond biol bit both brief briefly but by c ca came can cannot can’t cant cause causes certain certainly co com come comes contain containing contains cos could couldnt d date day did didn didn’t different do does doesn doesn’t doing don done don’t dont down downwards due during e each ed edu effect eg eight eighty either else elsewhere end ending enough especially et et-al etc even ever every everybody everyone everything everywhere ex except f far few ff fifth first five fix followed following follows for former formerly forth found four from further furthermore g game gave get gets getting give given gives giving go goes going gone gonna good got gotten great h had happens hardly has hasn hasn’t have haven haven’t having he hed hence her here hereafter hereby herein heres hereupon hers herself hes hey hi hid him himself his hither home how howbeit however hundred i id ie if i’ll im immediate immediately importance important in inc indeed index information instead into invention inward is isn isn’t it itd it’ll its itself i’ve j just k keep keeps kept keys kg km know known knows l ll largely last lately later latter latterly least less lest let lets like liked likely line little ll ’ll lol look looking looks lot ltd m made mate mainly make makes many may maybe me mean means meantime meanwhile merely mg might million miss ml more moreover most mostly mr mrs much mug must my myself n na name namely nay nd near nearly necessarily necessary need needs neither never nevertheless new next nine ninety no nobody non none nonetheless noone nor normally nos not noted nothing now nowhere o obtain obtained obviously of off often oh ok okay old omitted omg on once one ones only onto or ord other others otherwise ought our ours ourselves out outside over overall owing own p page pages part particular particularly past people per perhaps placed please plus poorly possible possibly potentially pp predominantly present previously primarily probably promptly proud provides put q que quickly quite qv r ran rather rd re readily really recent recently ref refs regarding regardless regards related relatively research respectively resulted resulting results right rt run s said same saw say saying says sec section see seeing seem seemed seeming seems seen self selves sent seven several shall she shed she’ll shes should shouldn shouldn’t show showed shown showns shows significant significantly similar similarly since six slightly so some somebody somehow someone somethan something sometime sometimes somewhat somewhere soon sorry specifically specified specify specifying state states still stop strongly sub substantially successfully such sufficiently suggest sup sure t take taken taking tbh tell tends th than thank thanks thanx that that’ll thats that’ve the their theirs them themselves then thence there thereafter thereby thered therefore therein there’ll thereof therere theres thereto thereupon there’ve these they theyd they’ll theyre they’ve thing things think this those thou though thoughh thousand throug through throughout thru thus til time tip to together too took toward towards tried tries truly try trying ts tweet twice two u un under unfortunately unless unlike unlikely until unto up upon ups ur us use used useful usefully usefulness uses using usually v value various ve ’ve very via viz vol vols vs w wanna want wants was wasn wasn’t way we wed welcome well we’ll went were weren weren’t we’ve what whatever what’ll whats when whence whenever where whereafter whereas whereby wherein wheres whereupon wherever whether which while whim whither who whod whoever whole who’ll whom whomever whos whose why widely will willing wish with within without won won’t words world would wouldn wouldn’t www x y yeah yes yet you youd you’ll your youre yours yourself yourselves you’ve z zero

Activities and Entities for Ubuntu Dialogue Corpus

Ubuntu activities:

accept, activate, add, ask, appoint, attach, backup, boot, check, choose, clean, click, comment, compare, compile, compress, change, affirm, connect, continue, administrate, copies, break, create, cut, debug, decipher, decompress, define, describe, debind, deattach, deactivate, download, adapt, eject, email, conceal, consider, execute, close, expand, expect, export, discover, correct, fold, freeze, get, deliver, go, grab, hash, import, include, install, interrupt, load, block, log, log-in, log-out, demote, build, clock, bind, more, mount, move, navigate, open, arrange, partition, paste, patch, plan, plug, post, practice, produce, pull, purge, push, put, queries, quote, look, reattach, reboot, receive, reject, release, remake, delete, name, replace, request, reset, resize, restart, retry, return, revert, reroute, scroll, send, set, display, shutdown, size, sleep, sort, split, come-up, store, signup, get-ahold-of, say, test, transfer, try, uncomment, de-expand, uninstall, unmount, unplug, unset, sign-out, update, upgrade, upload, use, delay, enter, support, prevent, loose, point, contain, access, share, buy, sell, help, work, mute, restrict, play, call, thank, burn, advice, force, repeat, stream, respond, browse, scan, restore, design, refresh, bundle, implement, programming, compute, touch, overheat, cause, affect, swap, format, rescue, zoomed, detect, dump, simulate, checkout, unblock, document, troubleshoot, convert, allocate, minimize, maximize, redirect, maintain, print, spam, throw, sync, contact, destroy

[8] Part of these were extracted from `https://github.com/defacto133/twitter-wordcloud-bot/blob/master/assets/stopwords-en.txt`

Ubuntu entities (excerpt):

ubuntu_7.04, dmraid, vnc4server, tasksel, aegis, mirage, system-config-audit, uif2iso, aumix, unrar, dell, hibernate, ucoded, finger, zoneminder, ucfg, macaddress, ia32-libs, synergy, aircrack-ng, pulseaudio, gnome, kid3, bittorrent, systemsettings, cups, finger, xchm, pan, uwidget, vnc-java, linux-source, ucommand.com, epiphany, avanade, onboard, uextended, substance, pmount, lilypond, proftpd, unii, jockey-common, aha, units, xrdp, mp3check, cruft, uemulator, ulivecd, amsn, ubuntu_5.10, acpidump, uadd-on, gpac, ifenslave, pidgin, soundconverter, kdelibs-bin, esmtp, vim, travel, smartdimmer, uactionscript, scrotwm, fbdesk, tulip, beep, nikto, wine, linux-image, azureus, vim, makefile, uuid, whiptail, alex, junior-arcade, libssl-dev, update-inetd, uextended, uaiglx, sudo, dump, lockout, overlay-scrollbar, xubuntu, mdk, mdm, mdf2iso, linux-libc-dev, sms, lm-sensors, dsl, lxde, dsh, smc, sdf, install-info, xsensors, gutenprint, sensors, ubuntu_13.04, atd, ata, fatrat, fglrx, equinix, atp, atx, libjpeg-dbg, umingw, update-inetd, firefox, devede, cd-r, tango, mixxx, uemulator, compiz, libpulse-dev, synaptic, ecryptfs, crawl, ugtk+, tree, perl, tree, ubuntu-docs, libsane, gnomeradio, ufilemaker, dyndns, libfreetype6, daemon, xsensors, vncviewer, vga, indicator-applet, nvidia-173, rsync, members, qemu, mount, rsync, macbook, gsfonts, synaptic, finger, john, cam, lpr, lpr, xsensors, lpr, lpr, screen, inotify, signatures, units, ushareware, ufraw, bonnie, nec, fstab, nano, bless, bibletime, irssi, ujump, foremost, nzbget, ssid, onboard, synaptic, branding, hostname, radio, hotwire, xebia, netcfg, xchat, irq, lazarus, pilot, ucopyleft, java-common, vm, ifplugd, ncmpcpp, irc, uclass, gnome, sram, binfmt-support, vuze, java-common, sauerbraten, adapter, login

Ubuntu commands:

alias, apt-get, aptitude, aspell, awk, basename, bc, bg, break, builtin, bzip2, cal, case, cat, cd, cfdisk, chgrp, chmod, chown, chroot, chkconfig, cksum, cmp, comm, command, continue, cp, cron, crontab, csplit, curl, cut, date, dc, dd, ddrescue, declare, df, diff, diff3, dig, dir, dircolors, dirname, dirs, dmesg, du, echo, egrep, eject, enable, env, eval, exec, exit, expect, expand, export, expr, false, fdformat, fdisk, fg, fgrep, file, find, fmt, fold, for, fsck, ftp, function, fuser, gawk, getopts, grep, groupadd, groupdel, groupmod, groups, gzip, hash, head, history, hostname, htop, iconv, id, if, ifconfig, ifdown, ifup, import, install, ip, jobs, join, kill, killall, less, let, link, ln, local, locate, logname, logout, look, lpc, lpr, lprm, ls, lsof, man, mkdir, mkfifo, mknod, more, most, mount, mtools, mtr, mv, mmv, nc, nl, nohup, notify-send, nslookup, open, op, passwd, paste, ping, pkill, popd, pr, printf, ps, pushd, pv, pwd, quota, quotacheck, quotactl, ram, rar, rcp, read, readonly, rename, return, rev, rm, rmdir, rsync, screen, scp, sdiff, sed, select, seq, set, shift, shopt, shutdown, sleep, slocate, sort, source, split, ssh, stat, strace, su, sudo, sum, suspend, sync, tail, tar, tee, test, time, timeout, times, touch, top, tput, traceroute, tr, true, tsort, tty, type, ulimit, umask, unalias, uname, unexpand, uniq, units, unrar, unset, unshar, until, useradd, userdel, usermod, users, uuencode, uudecode, vi, vmstat, wait, watch, wc, whereis, which, while, who, whoami, write, xargs, xdg-open, xz, yes, zip, admin, purge

10 Model Details

Training

All models were trained with a learning rate of 0.0002 or 0.0001, batches of size either 40 or size 80 and gradients are clipped at 1.0. We truncate the backpropagation to batches with 80 tokens We validate on the entire validation set every 5000 training batches. We choose almost identical hyperparameters for the Ubuntu and Twitter models, since the models appear to perform similarly w.r.t. different hyperparameters and since the statistics of the two datasets are comparable. We use the $20K$ most frequent words on Twitter and Ubuntu as the natural language vocabulary for all the models, and assign all words outside the vocabulary to a special unknown token symbol. For MrRNN, we use a coarse token vocabulary consisting of the $10K$ most frequent tokens in the coarse token sequences.

Generation

We compute the cost of each beam search (candidate response) as the log-likelihood of the tokens in the beam divided by the number of tokens it contains. The LSMT model performs better when the beam search is not allowed to generate the unknown token symbol, however even then it still performs worse than the HRED model across all metrics except for the command accuracy.

Baselines

Based on preliminary experiments, we found that a slightly different parametrization of the HRED baseline model worked better on Twitter. The *encoder* RNN has a bidirectional GRU RNN encoder, with 1000 hidden units for the forward and backward RNNs each, and a *context* RNN and a *decoder* RNN with 1000 hidden units each. Furthermore, the *decoder* RNN computes a 1000 dimensional real-valued vector for each hidden time step, which is multiplied with the output *context* RNN. The output is feed through a one-layer feed-forward neural network with hyperbolic tangent activation function, which the *decoder* RNN then conditions on.

11 Human Evaluation

All human evaluators either study or work in an English speaking environment, and have indicated that they have some experience using a Linux operating system. Before starting evaluators were shown one short annotated example with a brief explanation of how to give annotations. In particular, the evaluators were instructed to use the following reference in Figure 2.

Fluency	**Relevancy**
0 : Incomprehensible	**0** : None
1 : Disfluent English	**1** : Little relevance
2 : Non-Native English	**2** : Much relevance
3 : Good English	**3** : Most relevant
4 : Flawless English	**4** : All relevant

Figure 2: Fluency and relevancy reference table presented to human evaluators.

The 5 evaluators gave 1069 ratings in total. Table 6 shows the scores by category.

Table 6: Ubuntu human fluency and relevancy scores by rating category

	Fluency					**Relevancy**				
Rating Level / **Model**	0	1	2	3	4	0	1	2	3	4
HRED	3	11	21	50	49	68	22	19	19	4
HRED + Act.-Ent.	3	17	19	37	57	69	39	18	6	2
MrRNN Noun	1	2	8	52	71	51	45	24	10	4
MrRNN Act.-Ent	0	2	6	52	74	27	53	39	14	1

12 Model Examples

Table 7: Ubuntu model examples. The → token indicates a change of turn.

Context	Response
seriously... 50% cpu from xorg running just two apps(terminal and system monitor) i've been seeing this ever since i upgraded to 12.10 anyone else seeing this → I would do such kind of tests with ' top ' or ' something Enjoy the difference... → that is top saying 50% cpu from xorg → Ok. In that case I would try a window manager without composite for a try. 50% is too much.	**MrRNN Act. -Ent.:** Are you using XGL ? **MrRNN Noun:** What window manager are you using ? **HRED + Act.-Ent.:** I don't know how to help you with that, sorry. **HRED:** i have no idea how to do that, i just want to know how to do it **LSTM:** something ?
my linux seems to be stuck at rev 18... other systems in the house are as high as rev 34... any suggestions ? → what 's rev ? → the linux info says kernel linux says 3.5. something generic	**MrRNN Act. -Ent.:** how do i compile the kernel ? **MrRNN Noun:** the linux kernel is 2.6. something **HRED + Act.-Ent.:** how do i find out what version of ubuntu i have ? **HRED:** that 's not what you want to do **LSTM:** ? what do you mean ?
i need some SERIOUS help with fullscreen video playback. i already asked in #lubuntu, but i might have a better chance here : most video files either let the player hang or audio/video goes badly out of sync right away. i know that it does work with the same files and it even did so like two hours ago, but after some browsing and man page reading, the problem comes up again. i have NO idea where to look anymore nor what I did to ' reproduce ' the → even with vlc	**MrRNN Act. -Ent.:** works fine in vlc **MrRNN Noun:** do you use vlc ? **HRED + Act.-Ent.:** what video card do you have ? **HRED:** yeah, i know, but i don't know how to do that, but i don't know how to do that **LSTM:** ? what do you mean ?
I'm in the process of building a new desktop and I was wondering if there was any way to, using KVM, start or restart a Windows session when my girlfriend logs in. I have plenty of memory and my i5 3570 supports all the necessary virtualisation technologies. → you can install VirtualBox, and I'm sure there 's a command line method to run VB with a specified installed VM operating system, and you can make that command a startup command ending with " & "	**MrRNN Act. -Ent.:** have you installed virtualbox ? **MrRNN Noun:** are you trying to install ubuntu ? **HRED + Act.-Ent.:** I don't know how to do that, sorry. I don't know how to do that. **HRED:** thanks, I'll give it a shot. **LSTM:** thanks, I'll try that.
hello all, how can i setup two default gateways graphically in network manager ? → you can't have 2 default gateway on one device...? → well.. i can → hows that gonna work ?	**MrRNN Act. -Ent.:** what are you looking for ? **MrRNN Noun:** do you have a default gateway ? **HRED + Act.-Ent.:** what are you trying to do ? **HRED:** i want to set up a static ip, but i dont know how to set it up **LSTM:** ?
They are mounted via fstab → sure they are mounted ? → I've been using them all day :[p :P *	**MrRNN Act. -Ent.:** I see **MrRNN Noun:** something :/ something ls -ld something **HRED + Act.-Ent.:** I can mount it, but when I try to mount it, it says " mount : special device /dev/sda1 does not exist **HRED:** what do you mean by mount ? **LSTM:** ? what are you talking about ?

Table 8: Ubuntu model examples. The → token indicates a change of turn.

Context	Response
can you have gnome and KDE installed and use one or the other ? cough erin brainwaves	**MrRNN Act.-Ent.:** what are you looking for ? **MrRNN Noun:** gconf-editor **HRED + Act.-Ent.:** what do you want me to do ? **HRED:** yeah, i just did a fresh install of ubuntu, and it worked fine. **LSTM:** ?
Ugh so my mount points need to be in /media rather than /mnt to make them appear in the nautilus side-pane : → you could symlink /mnt to /media afaik	**MrRNN Act. -Ent.:** thank you :) **MrRNN Noun:** you need to mount it in /mnt or /media **HRED + Act.-Ent.:** if you don't know what you are trying to do, then you should be able to mount it manually, but I don't know how **HRED:** I don't know how to do that, I just want to be able to mount them in /media **LSTM:** what ?
How can i install ubuntu raring ? → ask in #ubuntu+1 → they are afk all of them → so ? doesn't mean its ontoppic here	**MrRNN Act. -Ent.:**can you just tell me how ? **MrRNN Noun:** join #ubuntu-offtopic **HRED + Act.-Ent.:** what are you trying to do ? **HRED:** i know, but i dont know how to install it **LSTM:** ?
How shall I reboot BIOS 100 time through script → i'm really curious why anyone would ever want to do that... → For BIOS stress testing	**MrRNN Act. -Ent.:**do you know how to check bios ? **MrRNN Noun:** How do I disable the bios ? **HRED + Act.-Ent.:** I don't want to reboot, I just want to be able to boot from a USB stick, but I don't want to have to do it every time I want to do it **HRED:** that 's not what you want to do, but you might want to ask in ##hardware **LSTM:** ?
what is the current version of wine in ubuntu ? i can see wine 1.5. However, when I try to install it via synaptic it wants to remove VLC ? → did you install vlc as a windows version ? → windows version on Ubuntu ?	**MrRNN Act. -Ent.:**what version of ff do you have installed ? **MrRNN Noun:** what version of ubuntu are you using ? **HRED + Act.-Ent.:** how do i install the latest version of wine ? i have the latest version, but i dont know how to install it **HRED:** yes **LSTM:** yes
Hey guys, I'm trying to install libreoffice from the repositories, but it 's throwing a wobbly. Can anyone help ? Looks like dependency problems, but when I attempt to resolve them it throws a wobbly at me. <url >→ try ' apt-get install -f ' yeah, it looks like you don't have java install You need to try to run ' apt-get install -f', just that string try ' apt-get purge libreoffice ' it looks like you had it installed previously → update : just realized I removed it earlier because it wasn't working. let me re-install. when I run ' apt-get purge libreoffice ' it just complains of unmet dependencies/ → but you were able to reinstall fine ? → no, it throws the errors seen in the pastebin at me → jwrigley, check to see if /var/lib/libreoffice/share/prereg exists → doesn't appear to exist when i cd to it in terminal → use apt-get via the command line tab completion helps a lot run ' apt-get purge libreoffice ' → oh, yeah i see now. sorry. now it is telling me to fix dependencies with apt-get install -f → awesome, so are you all set now ? → well libreoffice is still not working, so no.	**MrRNN Act. -Ent.:** I see. **MrRNN Noun:** how do I get rid of libreoffice ? **HRED + Act.-Ent.:** it says " E : Could not get lock /var/lib/dpkg/lock - open (11 Resource temporarily unavailable) E : Unable to locate package something " **HRED:** I don't know what that is, but I don't know how to fix it **LSTM:** is installed
Can anyone help get wireless working ? → whats the trouble ? → It doesnt show any wireless networks	**MrRNN Act. -Ent.:** It 's not working. **MrRNN Noun:** How do I get to the network manager ? **HRED + Act.-Ent.:** what is the name of your wireless card ? **HRED:** are you using ndiswrapper ? **LSTM:** what do you mean ?
Who can answer : " How can i get mac address of the server i just connected via pppoe ?" → try tcpdump ? tcpdump -e	**MrRNN Act. -Ent.:** what are you looking for ? **MrRNN Noun:** what is your mac address ? **HRED + Act.-Ent.:** what are you trying to do ? **HRED:** thanks, i'll try that **LSTM:** command not found