Exemplar Encoder-Decoder for Neural Conversation Generation

Gaurav Pandey, Danish Contractor, Vineet Kumar and Sachindra Joshi
IBM Research AI
New Delhi, India
{gpandey1, dcontrac, vineeku6, jsachind}@in.ibm.com

Abstract

In this paper we present the Exemplar Encoder-Decoder network (EED), a novel conversation model that learns to utilize *similar* examples from training data to generate responses. Similar conversation examples (context-response pairs) from training data are retrieved using a traditional TF-IDF based retrieval model. The retrieved responses are used to create exemplar vectors that are used by the decoder to generate the response. The contribution of each retrieved response is weighed by the similarity of corresponding context with the input context. We present detailed experiments on two large data sets and find that our method outperforms state of the art sequence to sequence generative models on several recently proposed evaluation metrics. We also observe that the responses generated by the proposed EED model are more informative and diverse compared to existing state-of-the-art method.

1 Introduction

With the availability of large datasets and the recent progress made by neural methods, variants of sequence to sequence learning (seq2seq) (Sutskever et al., 2014) architectures have been successfully applied for building conversational systems (Serban et al., 2016, 2017b). However, despite these methods being the state-of-the art frameworks for conversation generation, they suffer from problems such as lack of diversity in responses and generation of short, repetitive and uninteresting responses (Liu et al., 2016; Serban et al., 2016, 2017b). A large body of recent literature has focused on overcoming such challenges (Li et al., 2016a; Lowe et al., 2017).

In part, such problems arise as all information required to generate responses needs to be captured as part of the model parameters learnt from the training data. These model parameters alone may not be sufficient for generating natural conversations. Therefore, despite providing enormous amount of data, neural generative systems have been found to be ineffective for use in real world applications (Liu et al., 2016).

In this paper, we focus our attention on closed domain conversations. A characteristic feature of such conversations is that over a period of time, some conversation contexts[1] are likely to have occurred previously (Lu et al., 2017b). For instance, Table 1 shows some contexts from the Ubuntu dialog corpus. Each row presents an input dialog context with its corresponding gold response followed by a similar context and response seen in training data – as can be seen, contexts for "*installing dms*", "*sharing files*", "*blocking ufw ports*" have all occurred in training data. We hypothesize that being able to refer to training responses for previously seen similar contexts could be a helpful signal to use while generating responses.

In order to exploit this aspect of closed domain conversations we build our neural encoder-decoder architecture called the *Exemplar Encoder Decoder (EED)*, that learns to generate a response for a given context by exploiting *similar* contexts from training conversations. Thus, instead of having the seq2seq model learn patterns of language only from aligned parallel corpora, we assist the model by providing it *closely* related (*similar*) samples from the training data that it can *refer* to while generating text.

Specifically, given a context c, we retrieve a set

[1] We use the phrase "dialog context", "conversation context" and "context" interchangeably throughout the paper.

Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers), pages 1329–1338
Melbourne, Australia, July 15 - 20, 2018.

	Input Context	Gold Response		Similar Context in training data	Associated Response
U1 U2	if you want autologin install a dm of some sort what is a dm	lightdm, **gdm**, **kdm**, **xdm**, slim, etc.	U1 U2	if you're running a dm, it will probably restart x whats a dm?	e.g. **gdm**, **kdm**, **xdm**
U1 U2	is it possible to share a file in one user's home directory with another user? if you set permissions (to 'group','other' or with an acl)	so **chmod** 777 should do it, right?	U1 U2	howto set right permission for my home directory? but which is the syntax to set permission for my user in my home user directory ?	**chmod** and chown? u mean that sintax
U1 U2	is there a way to block all ports in ufw and only allow the ports that have been allowed? try to get familiar with configuring iptables	do i need to use iptables in order to use **ufw?**	U1 U2	is ufw blocking connections to all ports by default? no, all ports are open by default.	how do i block all ports with **ufw**?
U1 U2	how do i upgrade on php beyond 5.3.2 on ubuntu using apt-get ? ? ? this version is a bit old which version of ubuntu are you using?	lucid, **10.04 ubuntu** 10.04.4 lts	U1 U2 U1	hello!, how can i upgrade apt-get?(i have version 0.7.9 installed but i need to update to latest) sudo apt-get upgrade apt-get what version of ubuntu do you have?	I'm using **ubuntu** server **10.04** 64

Table 1: Sample input contexts and corresponding gold responses from Ubuntu *validation* dataset along with similar contexts seen in training data and their corresponding responses. We refer to training data as training data for the Ubuntu corpus. The highlighted words are common between the gold response and the exemplar response.

of context-response pairs $(c^{(k)}, r^{(k)})$, $1 \leq k \leq K$ using an inverted index of training data. We create an *exemplar vector* $e^{(k)}$ by encoding the response $r^{(k)}$ (also referred to as exemplar response) along with an encoded representation of the current context c. We then learn the *importance* of each exemplar vector $e^{(k)}$ based on the likelihood of it being able to generate the ground truth response. We believe that $e^{(k)}$ may contain information that is helpful in generating the response. Table 1 highlights the words in exemplar responses that appear in the ground truth response as well.

Contributions: We present a novel Exemplar Encoder-Decoder (EED) architecture that makes use of similar conversations, fetched from an index of training data. The retrieved context-response pairs are used to create *exemplar* vectors which are used by the decoder in the EED model, to learn the importance of training context-response pairs, while generating responses. We present detailed experiments on the publicly benchmarked Ubuntu dialog corpus data set (Lowe et al., 2015) as well a large collection of more than 127,000 technical support conversations. We compare the performance of the EED model with the existing state of the art generative models such as HRED (Serban et al., 2016) and VHRED (Serban et al., 2017b). We find that our model out-performs these models on a wide variety of metrics such as the recently proposed *Activity Entity* metrics (Serban et al., 2017a) as well as Embedding-based metrics (Lowe et al., 2015). In addition, we present qualitative insights into our results and we find that exemplar based responses are more informative and diverse.

The rest of the paper is organized as follows. Section 2 briefly describes the recent works in neural dialogue generation The details of the proposed EED model for dialogue generation are described in detail in Section 3. In Section 4, we describe the datasets as well as the details of the models used during training. We present quantitative and qualitative results of EED model in Section 5.

2 Related Work

In this section, we compare our work against other data-driven end-to-end conversation models. End-to-end conversation models can be further classified into two broad categories — *generation* based models and *retrieval* based models.

Generation based models cast the problem of dialogue generation as a sequence to sequence learning problem. Initial works treat the entire context as a single long sentence and learn an encoder-decoder framework to generate response word by word (Shang et al., 2015; Vinyals and Le, 2015). This was followed by work that models context better by breaking it into conversation history and last utterance (Sordoni et al., 2015b). Context was further modeled effectively by using a hierarchical encoder decoder (HRED) model which first learns a vector representation of each utterance and then combines these representations to learn vector representation of context (Serban et al., 2016). Later, an alternative hierarchical model called VHRED (Serban et al., 2017b) was proposed, where generated responses were conditioned on latent variables. This leads to more in-

formative responses and adds diversity to response generation. Models that explicitly incorporate diversity in response generation have also been studied in literature (Li et al., 2016b; Vijayakumar et al., 2016; Cao and Clark, 2017; Zhao et al., 2017).

Our work differs from the above as none of these above approaches utilize similar conversation contexts observed in the training data explicitly.

Retrieval based models on the other hand treat the conversation context as a query and obtain a set of responses using information retrieval (IR) techniques from the conversation logs (Ji et al., 2014). There has been further work where the responses are further ranked using a deep learning based model (Yan et al., 2016a,b; Qiu et al., 2017). On the other hand of the spectrum, end-to-end deep learning based rankers have also been employed to generate responses (Wu et al., 2017; Henderson et al., 2017). Recently a framework has also been proposed that uses a discriminative dialog network that ranks the candidate responses received from a response generator network and trains both the networks in an end to end manner (Lu et al., 2017a).

In contrast to the above models, we use the input contexts as well as the retrieved responses for generating the final responses. Contemporaneous to our work, a generative model for machine translation that employs retrieved translation pairs has also been proposed (Gu et al., 2017). We note that while the underlying premise of both the papers remains the same, the difference lies in the mechanism of incorporating the retrieved data.

3 Exemplar Encoder Decoder

3.1 Overview

A conversation consists of a sequence of utterances. At a given point in the conversation, the utterances expressed prior to it are jointly referred to as the context. The utterance that immediately follows the context is referred to as the response. As discussed in Section 1, given a conversational context, we wish to to generate a response by utilizing similar context-response pairs from the training data. We retrieve a set of K exemplar context-response pairs from an inverted index created using the training data in an off-line manner. The input and the retrieved context-response pairs are then fed to the Exemplar Encoder Decoder (EED) network. A schematic illustration of the EED network is presented in Figure 1. The EED encoder combines the input context and the retrieved responses to create a set of *exemplar vectors*. The EED decoder then uses the exemplar vectors based on the similarity between the input context and retrieved contexts to generate a response. We now provide details of each of these modules.

3.2 Retrieval of Similar Context-Response Pairs

Given a large collection of conversations as $(context,\ response)$ pairs, we index each response and its corresponding context in $tf - idf$ vector space. We further extract the last *turn* of a conversation and index it as an additional attribute of the context-response document pairs so as to allow directed queries based on it.

Given an input context c, we construct a query that weighs the last utterance in the context twice as much as the rest of the context and use it to retrieve the top-k similar context-response pairs from the index based on a BM25 (Robertson et al., 2009) retrieval model. These retrieved pairs form our exemplar context-response pairs $(c^{(k)}, r^{(k)}),\ 1 \leq k \leq K$.

3.3 Exemplar Encoder Network

Given the exemplar pairs $(c^{(k)}, r^{(k)}),\ 1 \leq k \leq K$ and an input context-response pair (c, r), we feed the input context c and the exemplar contexts $c^{(1)}, \ldots, c^{(K)}$ through an encoder to generate the embeddings as given below:

$$c_e = Encode_c(c)$$
$$c_e^{(k)} = Encode_c(c^{(k)}),\ 1 \leq k \leq K$$

Note that we do not constrain our choice of encoder and that any parametrized differentiable architecture can be used as the encoder to generate the above embeddings. Similarly, we feed the exemplar responses $r^{(1)}, \ldots, r^{(K)}$ through a response encoder to generate response embeddings $r_e^{(1)}, \ldots, r_e^{(K)}$, that is,

$$r_e^{(k)} = Encode_r(r^{(k)}),\ 1 \leq k \leq K \qquad (1)$$

Next, we concatenate the exemplar response encoding $r_e^{(k)}$ with an encoded representation of current context c_e as shown in equation 2 to create the exemplar vector $e^{(k)}$. This allows us to include in-

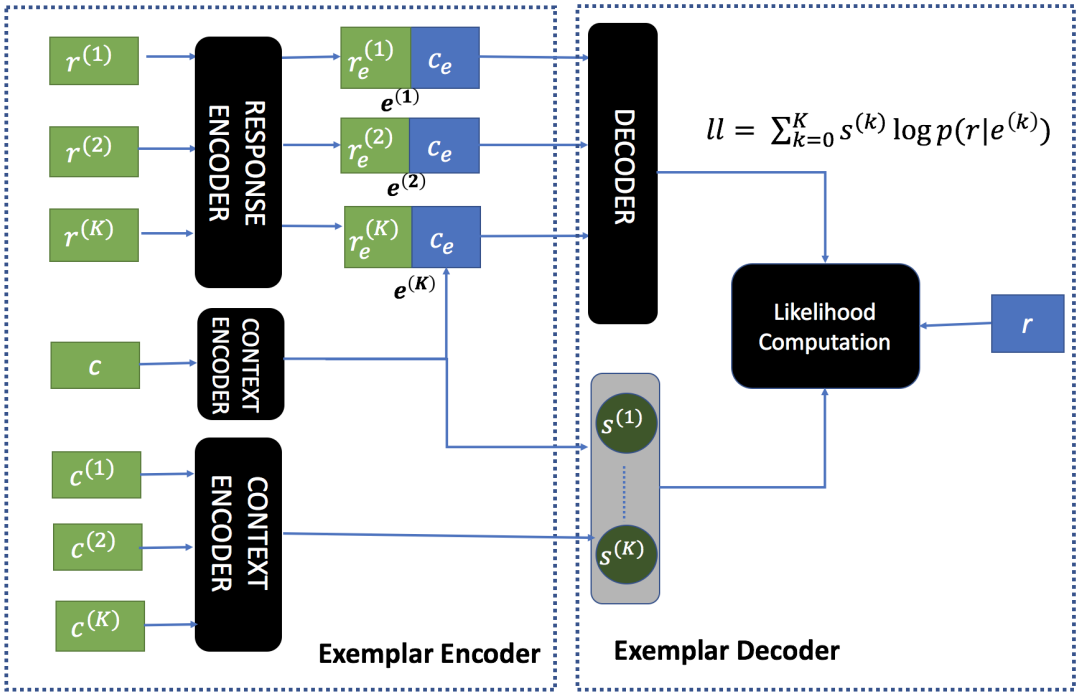

Figure 1: A schematic illustration of the EED network. The input context-response pair is (c, r), while the exemplar context-response pairs are $(c^{(k)}, r^{(k)})$, $1 \leq k \leq K$.

formation about similar responses along with the encoded input context representation.

$$e^{(k)} = [c_e; r_e^{(k)}],\ 1 \leq k \leq K \tag{2}$$

The exemplar vectors $e^{(k)}$, $1 \leq k \leq K$ are further used by the decoder for generating the ground truth response as described in the next section.

3.4 Exemplar Decoder Network

Recall that we want the exemplar responses to help generate the responses based on how similar the corresponding contexts are with the input context. More similar an exemplar context is to the input context, higher should be its effect in generating the response. To this end, we compute the similarity scores $s^{(k)}$, $1 \leq k \leq K$ using the encodings computed in Section 3.3 as shown below.

$$s^{(k)} = \frac{\exp(c_e^{\mathrm{T}} c_e^{(k)})}{\sum_{l=1}^{K} \exp(c_e^{\mathrm{T}} c_e^{(l)})} \tag{3}$$

Next, each exemplar vector $e^{(k)}$ computed in Section 3.3, is fed to a decoder, where the decoder is responsible for predicting the ground truth response from the exemplar vector. Let $p^{dec}(r|e^{(k)})$ be the distribution of generating the ground truth response given the exemplar embedding. The objective function to be maximized, is expressed as a function of the scores $s^{(k)}$, the decoding distribution p^{dec} and the exemplar vectors $e^{(k)}$ as shown below:

$$ll = \sum_{k=1}^{K} s^{(k)} \log p^{dec}(r|e^{(k)}) \tag{4}$$

Note that we weigh the contribution of each exemplar vector to the final objective based on how similar the corresponding context is to the input context. Moreover, the similarities are differentiable function of the input and hence, trainable by back propagation. The model should learn to assign higher similarities to the exemplar contexts, whose responses are helpful for generating the correct response.

The model description uses encoder and decoder networks that can be implemented using any differentiable parametrized architecture. We discuss our choices for the encoders and decoder in the next section.

3.5 The Encoders and Decoder

In this section, we discuss the various encoders and the decoder used by our model. The conversation context consists of an ordered sequence of utterances and each utterance can be further viewed as a sequence of words. Thus, context can be viewed as having multiple levels of

hierarchies—at the word level and then at the utterance (sentence) level. We use a hierarchical recurrent encoder—popularly employed as part of the HRED framework for generating responses and query suggestions (Sordoni et al., 2015a; Serban et al., 2016, 2017b). The word-level encoder encodes the vector representations of words of an utterance to an utterance vector. Finally, the utterance-level encoder encodes the utterance vectors to a context vector.

Let $(\mathbf{u}_1, \ldots, \mathbf{u}_N)$ be the utterances present in the context. Furthermore, let $(w_{n1}, \ldots, w_{nM_n})$ be the words present in the n^{th} utterance for $1 \leq n \leq N$. For each word in the utterance, we retrieve its corresponding embedding from an embedding matrix. The word embedding for w_{nm} will be denoted as w_{enm}. The encoding of the n^{th} utterance can be computed iteratively as follows:

$$h_{nm} = f_1(h_{nm-1}, w_{enm}),\ 1 \leq m \leq M_n \quad (5)$$

We use an LSTM (Hochreiter and Schmidhuber, 1997) to model the above equation. The last hidden state h_{nM_n} is referred to as the utterance encoding and will be denoted as h_n.

The utterance-level encoder takes the utterance encodings $h_1, \ldots, h_N$ as input and generates the encoding for the context as follows:

$$c_{en} = f_2(c_{en-1}, h_n),\ 1 \leq n \leq N \quad (6)$$

Again, we use an LSTM to model the above equation. The last hidden state c_{eN} is referred to as the context embedding and is denoted as c_e.

A single level LSTM is used for embedding the response. In particular, let $(w_1, \ldots, w_M)$ be the sequence of words present in the response. For each word w, we retrieve the corresponding word embedding w_e from a word embedding matrix. The response embedding is computed from the word embeddings iteratively as follows:

$$r_{em} = g(r_{em-1}, w_{em}),\ 1 \leq m \leq M \quad (7)$$

Again, we use an LSTM to model the above equation. The last hidden state r_{em} is referred to as the response embedding and is denoted as r_e.

4 Experimental Setup

4.1 Datasets

4.1.1 Ubuntu Dataset

We conduct experiments on Ubuntu Dialogue Corpus (Lowe et al., 2015)(v2.0)[2]. Ubuntu dialogue corpus has about 1M context response pairs along with a label. The label value 1 indicates that the response associated with a context is the correct response and is incorrect otherwise. As we are only interested in positive labeled data we work with $label = 1$. Table 2 depicts some statistics for the dataset.

	Size
Training Pairs	499,873
Validation Pairs	19,560
Test Pairs	18,920
$\|V\|$	538,328

Table 2: Dataset statistics for Ubuntu Dialog Corpus v2.0 (Lowe et al., 2015), where $|V|$ represents the size of vocabulary.

4.1.2 Tech Support Dataset

We also conduct our experiments on a large technical support dataset with more than 127K conversations. We will refer to this dataset as Tech Support dataset in the rest of the paper. Tech Support dataset contains conversations pertaining to an employee seeking assistance from an agent (technical support) — to resolve problems such as password reset, software installation/licensing, and wireless access. In contrast to Ubuntu dataset, this dataset has clearly two distinct users — employee and agent. In our experiments we model the *agent* responses only.

For each conversation in the tech support data, we sample context and response pairs to create a dataset similar to the Ubuntu dataset format. Note that multiple context-response pairs can be generated from a single conversation. For each conversation, we sample 25% of the possible context-response pairs. We create validation pairs by selecting 5000 conversations randomly and sampling context response pairs). Similarly, we create test pairs from a different subset of 5000 conversations. The remaining conversations are used to

[2] https://github.com/rkadlec/ubuntu-ranking-dataset-creator

create training context-response pairs. Table 3 depicts some statistics for this dataset:

	Size
Conversations	127,466
Training Pairs	204,808
Validation Pairs	8,738
Test Pairs	8,756
$\|V\|$	293,494

Table 3: Dataset statistics for Tech Support dataset.

4.2 Model and Training Details

The EED and HRED models were implemented using the PyTorch framework (Paszke et al., 2017). We initialize the word embedding matrix as well as the weights of context and response encoders from the standard normal distribution with mean 0 and variance 0.01. The biases of the encoders and decoder are initialized with 0. The word embedding matrix is shared by the context and response encoders. For Ubuntu dataset, we use a word embedding size of 600, whereas the size of the hidden layers of the LSTMs in context and response encoders and the decoder is fixed at 1200. For Tech support dataset, we use a word embedding size of 128. Furthermore, the size of the hidden layers of the multiple LSTMs in context and response encoders and the decoder is fixed at 256. A smaller embedding size was chosen for the Tech Support dataset since we observed much less diversity in the responses of the Tech Support dataset as compared to Ubuntu dataset.

Two different encoders are used for encoding the input context (not shown in Figure 1 for simplicity). The output of the first context encoder is concatenated with the exemplar response vectors to generate exemplar vectors as detailed in Section 3.3. The output of the second context encoder is used to compute the scoring function as detailed in Section 3.4. For each input context, we retrieve 5 similar context-response pairs for Ubuntu dataset and 3 context-response pairs for Tech support dataset using the tf-idf mechanism discussed in Section 3.2.

We use the Adam optimizer (Kingma and Ba, 2014) with a learning rate of $1e-4$ for training the model. A batch size of 20 samples was used during training. In order to prevent overfitting, we use early stopping with log-likelihood on validation set as the stopping criteria. In order to generate the samples using the proposed EED model, we identify the exemplar context that is most similar to the input context based on the learnt scoring function discussed in Section 3.4. The corresponding exemplar vector is fed to the decoder to generate the response. The samples are generated using a beam search with width 5. The average per-word log-likelihood is used to score the beams.

5 Results & Evaluation

5.1 Quantitative Evaluation

5.1.1 Activity and Entity Metrics

A traditional and popular metric used for comparing a generated sentence with a ground truth sentence is BLEU (Papineni et al., 2002) and is frequently used to evaluate machine translation. The metric has also been applied to compute scores for predicted responses in conversations, but it has been found to be less indicative of actual performance (Liu et al., 2016; Sordoni et al., 2015a; Serban et al., 2017a), as it is extremely sensitive to the exact words in the ground truth response, and gives equal importance to stop words/phrases and informative words.

Serban et al. (2017a) recently proposed a new set of metrics for evaluating dialogue responses for the Ubuntu corpus. It is important to highlight that these metrics have been specifically designed for the Ubuntu corpus and evaluate a generated response with the ground truth response by comparing the coarse level representation of an utterance (such as entities, activities, Ubuntu OS commands). Here is a brief description of each metric:

- **Activity**: Activity metric compares the activities present in a predicted response with the ground truth response. Activity can be thought of as a verb. Thus, all the verbs in a response are mapped to a set of manually identified list of 192 verbs.

- **Entity**: This compares the technical entities that overlap with the ground truth response. A total of 3115 technical entities is identified using public resources such as Debian package manager APT.

	Activity			Entity			Tense	Cmd
Model	**P**	**R**	**F1**	**P**	**R**	**F1**	**Acc.**	**Acc.**
LSTM*	1.7	1.03	1.18	1.18	0.81	0.87	14.57	94.79
VHRED*	**6.43**	4.31	4.63	3.28	2.41	2.53	20.2	92.02
HRED*	5.93	4.05	4.34	2.81	2.16	2.22	22.2	92.58
EED	6.42	**4.77**	**4.87**	**3.8**	**2.91**	**2.99**	**31.73**	**95.06**

Table 4: Activity & Entity metrics for the Ubuntu corpus. LSTM*, HRED* & VHRED* as reported by Serban et al. (2017a).

- **Tense**: This measure compares the time tense of ground truth with predicted response.

- **Cmd**: This metric computes accuracy by comparing commands identified in ground truth utterance with a predicted response.

Table 4 compares our model with other recent generative models (Serban et al., 2017a) — LSTM (Shang et al., 2015), HRED (Serban et al., 2016) & VHRED (Serban et al., 2017b).We do not compare our model with Multi-Resolution RNN (MRNN) (Serban et al., 2017a), as MRNN explicitly utilizes the activities and entities during the generation process. In contrast, the proposed EED model and the other models used for comparison are agnostic to the activity and entity information. We use the standard script[3] to compute the metrics.

The EED model scores better than generative models on almost all of the metrics, indicating that we generate more *informative* responses than other state-of-the-art generative based approaches for Ubuntu corpus. The results show that responses associated with similar contexts may contain the activities and entities present in the ground truth response, and thus help in response generation. This is discussed further in Section 5.2. Additionally, we compared our proposed EED with a retrieval only baseline. The retrieval baseline achieves an activity F1 score of 4.23 and entity F1 score of 2.72 compared to 4.87 and 2.99 respectively achieved by our method on the Ubuntu corpus.

The Tech Support dataset is not evaluated using the above metrics, since activity and entity information is not available for this dataset.

[3]https://github.com/julianser/Ubuntu-Multiresolution-Tools/blob/master/ActEntRepresentation/eval_file.sh

5.1.2 Embedding Metrics

Embedding metrics (Lowe et al., 2017) were proposed as an alternative to word by word comparison metrics such as BLEU. We use pre-trained Google news word embeddings[4] similar to Serban et al. (2017b), for easy reproducibility as these metrics are sensitive to the word embeddings used. The three metrics of interest utilize the word vectors in ground truth response and a predicted response and are discussed below:

- **Average**: Average word embedding vectors are computed for the candidate response and ground truth. The cosine similarity is computed between these averaged embeddings. High similarity gives as indication that ground truth and predicted response have similar words.

- **Greedy**: Greedy matching score finds the most similar word in predicted response to ground truth response using cosine similarity.

- **Extrema**: Vector extrema score computes the maximum or minimum value of each dimension of word vectors in candidate response and ground truth.

Of these, the embedding average metric is the most reflective of performance for our setup. The extrema representation, for instance, is very sensitive to text length and becomes ineffective beyond single length sentences(Forgues et al., 2014). We use the publicly available script[5] for all our computations. As the test outputs for HRED are not available for Technical Support dataset, we use our

[4]GoogleNews-vectors-negative300.bin from https://code.google.com/archive/p/word2vec/

[5]https://github.com/julianser/hed-dlg-truncated/blob/master/Evaluation/embedding_metrics.py

	Ubuntu			Tech Support		
Model	**Average**	**Greedy**	**Extrema**	**Average**	**Greedy**	**Extrema**
HRED	0.5131	0.4103	**0.3263**	0.6372	0.5100	0.4355
EED	**0.5528**	**0.4172**	0.3134	**0.6420**	**0.5125**	**0.4356**

Table 5: Embedding Metrics (Lowe et al., 2015) for Ubuntu and Technical Support Corpus.

	Ubuntu			Tech Support		
Model	**distinct-1**	**distinct-2**	**distinct-3**	**distinct-1**	**distinct-2**	**distinct-3**
HRED	1154	8668	22417	645	3510	6365
EED	**1173**	**9572**	**29447**	**732**	**4149**	**7877**

Table 6: The number of unique tokens, token-pairs and token-triplets for Ubuntu and Technical Support Corpus.

	Input Context (C) and exemplar response (R)	**Generated/Ground Truth Responses**
1	**C:** i want to check ntfs file systems. is fsck ok with ntfs. → no , you 'd need to use chkdsk from windows. → **R:** i already symlinked ntfsfix to fsck. **ntfs** and am scanning the unmounted devices	**HRED:** i know , but i dont know how to do that **VHRED:** how can i do that ? **EED:** i am trying to use **ntfs** 3g i have a windows partition. **GT:** i dont have windows installed
2	**C:** anyone help me ? → seems that its detecting grub , have you tried checking continue without installing grub ? **R:** try the live **cd**... you ca n't boot into linux unless **grub install**	**HRED:** yes i did , but i have no idea how to fix it **VHRED:** yes i did. i dont know how to do it **EED:** nope not yet. **grub install cd** is not working **GT:** no i did not.
3	**C:** any ideas on my mplayer problem ? → reinstall. apt-get purge mplayer then reinstall **R:** but half my system depends on this **package** ...	**HRED:** I did that , but it didn't work **VHRED:** I dont know how to do that **EED:** does n t work. **package** does n t work **GT:** that does n't seem as a solution
4	**C:** any of you guys in here know how to take a package on launchpad and recompile it and submit to your own ppa ? https : //launchpad.net/ mojocode/+archive /ppa ¡ ¡ i 'm trying to take the nginx package and install the rtmp module → could n't you download the source and just commit it to your own ? **R:** i **need** prop . drivers	**HRED:** I don't know how to do that , I just want to know if there is a way to do it **VHRED:** no , I just want to make it so I can use it **EED:** i **need** to get the source package. package manager is not available **GT:** so i figured how how to copy into my private ppa . any idea how to download those source files maybe and make modifications to resubmit and recompile ? https : //launchpad.net / blendedbyus/+archive/master/+packages
5	**C:** hi people , can you help me uninstall a package ? i did dpkg -i virtualbox-4.2_4.2.2-81494 ubuntu precise _amd64.deb. how do i uninstall it , i ca n't find the package from apt-get → sudo apt-get purge virtualbox should work fine → package virtualbox is not installed , so not removed → does n't work : (**R: sudo dpkg** – remove – force < package >	**HRED:** what version of ubuntu ? **VHRED:** did you try sudo apt-get install virtualbox-ose **EED: sudo dpkg** reconfigure package package manager **GT:** if you ca n't find the package ; sudo aptitude search virtualbox

Table 7: Contexts, exemplar responses and responses generated by HRED, VHRED and the proposed EED model. We use the published responses for HRED and VHRED. GT indicates the ground truth response. The change of turn is indicated by →. The highlighted words in **bold** are common between the exemplar response and the response predicted by EED.

own implementation of HRED. Table 5 compares our model with HRED, and depicts that our model scores better on all metrics for Technical Support dataset, and on majority of the metrics for Ubuntu dataset.

We note that the improvement achieved by the

EED model on activity and entity metrics are much more significant than those on embedding metrics. This suggests that the EED model is better able to capture the specific information (objects and actions) present in the conversations.

Finally, we evaluate the diversity of the generated responses for EED against HRED by counting the number of unique tokens, token-pairs and token-triplets present in the generated responses on Ubuntu and Tech Support dataset. The results are shown in Table 6. As can be observed, the responses in EED have a larger number of distinct tokens, token-pairs and token-triplets than HRED, and hence, are arguably more diverse.

5.2 Qualitative Evaluation

Table 7 presents the responses generated by HRED, VHRED and the proposed EED for a few selected contexts along with the corresponding similar exemplar responses. As can be observed from the table, the responses generated by EED tend to be more specific to the input context as compared to the responses of HRED and VHRED. For example, in conversations 1 and 2 we find that both HRED and VHRED generate simple generic responses whereas EED generates responses with additional information such as the type of disk partition used or a command not working. This is also confirmed by the quantitative results obtained using activity and entity metrics in the previous section. We further observe that the exemplar responses contain informative words that are utilized by the EED model for generating the responses as highlighted in Table 7.

6 Conclusions

In this work, we propose a deep learning method, Exemplar Encoder Decoder (EED), that given a conversation context uses similar contexts and corresponding responses from training data for generating a response. We show that by utilizing this information the system is able to outperform state of the art generative models on publicly available Ubuntu dataset. We further show improvements achieved by the proposed method on a large collection of technical support conversations.

While in this work, we apply the exemplar encoder decoder network on conversational task, the method is generic and could be used with other tasks such as question answering and machine translation. In our future work we plan to extend the proposed method to these other applications.

Acknowledgements

We are grateful to the anonymous reviewers for their comments that helped in improving the paper.

References

Kris Cao and Stephen Clark. 2017. Latent variable dialogue models and their diversity. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, volume 2, pages 182–187.

Gabriel Forgues, Joelle Pineau, Jean-Marie Larchevêque, and Réal Tremblay. 2014. Bootstrapping dialog systems with word embeddings. In *Nips, modern machine learning and natural language processing workshop*, volume 2.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor OK Li. 2017. Search engine guided non-parametric neural machine translation. *arXiv preprint arXiv:1705.07267*.

Matthew Henderson, Rami Al-Rfou', Brian Strope, Yun-Hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient natural language response suggestion for smart reply. *CoRR*, abs/1705.00652.

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

Zongcheng Ji, Zhengdong Lu, and Hang Li. 2014. An information retrieval approach to short text conversation. *CoRR*, abs/1408.6988.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 110–119.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016b. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119.

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023*.

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

Ryan Thomas Lowe, Nissan Pow, Iulian Vlad Serban, Laurent Charlin, Chia-Wei Liu, and Joelle Pineau. 2017. Training end-to-end dialogue systems with the ubuntu dialogue corpus. *Dialogue & Discourse*, 8(1):31–65.

Jiasen Lu, Anitha Kannan, Jianwei Yang, Devi Parikh, and Dhruv Batra. 2017a. Best of both worlds: Transferring knowledge from discriminative learning to a generative visual dialog model. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 314–324.

Yichao Lu, Phillip Keung, Shaonan Zhang, Jason Sun, and Vikas Bhardwaj. 2017b. A practical approach to dialogue response generation in closed domains. *arXiv preprint arXiv:1703.09439*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Minghui Qiu, Feng-Lin Li, Siyu Wang, Xing Gao, Yan Chen, Weipeng Zhao, Haiqing Chen, Jun Huang, and Wei Chu. 2017. Alime chat: A sequence to sequence and rerank based chatbot engine. In *ACL*.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Iulian Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*.

Iulian Vlad Serban, Tim Klinger, Gerald Tesauro, Kartik Talamadupula, Bowen Zhou, Yoshua Bengio, and Aaron C Courville. 2017a. Multiresolution recurrent neural networks: An application to dialogue response generation. In *AAAI*, pages 3288–3294.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017b. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*, pages 3295–3301.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *ACL*.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015a. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 553–562. ACM.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and William B. Dolan. 2015b. A neural network approach to context-sensitive generation of conversational responses. In *HLT-NAACL*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2016. Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424*.

Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. *CoRR*, abs/1506.05869.

Yu Wu, Wei Wu, Chen Xing, Can Xu, Zhoujun Li, and Ming Zhou. 2017. A sequential matching framework for multi-turn response selection in retrieval-based chatbots. *CoRR*, abs/1710.11344.

Rui Yan, Yiping Song, and Hua Wu. 2016a. Learning to respond with deep neural networks for retrieval-based human-computer conversation system. In *SIGIR*.

Rui Yan, Yiping Song, Xiangyang Zhou, and Hua Wu. 2016b. "shall i be your chat companion?": Towards an online human-computer conversation system. In *CIKM*.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 654–664.