Response Generation by Context-aware Prototype Editing

Yu Wu[†◇], **Furu Wei**[‡], **Shaohan Huang**[‡], **Yunli Wang**[†], **Zhoujun Li**[†◇], **Ming Zhou**[‡]
[†]State Key Lab of Software Development Environment, Beihang University, Beijing, China
[◇] Authors are supported by AdeptMind Scholarship
[‡] Microsoft Research, Beijing, China
{wuyu,wangyunli,lizj}@buaa.edu.cn {fuwei, shaohanh, mingzhou}@microsoft.com

Abstract

Open domain response generation has achieved remarkable progress in recent years, but sometimes yields short and uninformative responses. We propose a new paradigm, prototype-then-edit for response generation, that first retrieves a prototype response from a pre-defined index and then edits the prototype response according to the differences between the prototype context and current context. Our motivation is that the retrieved prototype provides a good start-point for generation because it is grammatical and informative, and the post-editing process further improves the relevance and coherence of the prototype. In practice, we design a context-aware editing model that is built upon an encoder-decoder framework augmented with an editing vector. We first generate an edit vector by considering lexical differences between a prototype context and current context. After that, the edit vector and the prototype response representation are fed to a decoder to generate a new response. Experiment results on a large scale dataset demonstrate that our new paradigm significantly increases the relevance, diversity and originality of generation results, compared to traditional generative models. Furthermore, our model outperforms retrieval-based methods in terms of relevance and originality.

Introduction

In recent years, non-task oriented chatbots focused on responding to humans intelligently on a variety of topics, have drawn much attention from both academia and industry. Existing approaches can be categorized into generation-based methods (Shang, Lu, and Li 2015; Vinyals and Le 2015; Serban et al. 2016; Sordoni et al. 2015; Serban et al. 2017) which generate a response from scratch, and retrieval-based methods (Hu et al. 2014; Lowe et al. 2015; Yan, Song, and Wu 2016; Zhou et al. 2016; Wu et al. 2017) which select a response from an existing corpus. Since retrieval-based approaches are severely constrained by a pre-defined index, generative approaches become more and more popular in recent years. Traditional generation-based approaches, however, do not easily generate long, diverse and informative responses, which is referred to as "safe response" problem (Li et al. 2016a).

To address this issue, we propose a new paradigm, prototype-then-edit, for response generation. Our motivations include: 1) human-written responses, termed as "prototypes response", are informative, diverse and grammatical which do not suffer from short and generic issues. Hence, generating responses by editing such prototypes is able to alleviate the "safe response" problem. 2) Some retrieved prototypes are not relevant to the current context, or suffer from a privacy issue. The post-editing process can partially solve these two problems. 3) Lexical differences between contexts provide an important signal for response editing. If a word appears in the current context but not in the prototype context, the word is likely to be inserted into the prototype response in the editing process.

Context (c)	My friends and I ~~went to some~~ vegan place for ~~dessert~~ yesterday.
Prototype context (c')	My friends and I had Tofu and vegetables at a vegan place nearby yesterday.
Prototype response (r')	**Raw green vegetables** are very **beneficial** for your health.
Revised response (r)	**Desserts** are very **bad** for your health.

Table 1: An example of context-aware prototypes editing. Underlined words mean they do not appear in the original context, while ~~words with strikethrough~~ mean they are not in the prototype context. Words in bold represent they are modified in the revised response.

Inspired by this idea, we formulate the response generation process as follows. Given a conversational context c, we first retrieve a similar context c' and its associated response r' from a pre-defined index, which are called prototype context and prototype response respectively. Then, we calculate an edit vector by concatenating the weighted average results of insertion word embeddings (words in prototype context but not in current context) and deletion word embeddings (words in current context but not in prototype context). After that, we revise the prototype response conditioning on the edit vector. We further illustrate how our idea works with an example in Table 1. It is obvious that the major difference between c and c' is what the speaker eats, so the phrase "raw green vegetables" in r' should be replaced by "desserts" in order to adapt to the current context c. We hope that the

decoder language model could remember the collocation of "desserts" and "bad for health", so as to replace "beneficial" with "bad" in the revised response. The new paradigm does not only inherits the fluency and informativeness advantages from retrieval results, but also enjoys the flexibility of generation results. Hence, our edit-based model is better than previous retrieval-based and generation-based models. The edit-based model can solve the "safe response" problem of generative models by leveraging existing responses, and is more flexible than retrieval-based models, because it does not highly depend on the index and is able to edit a response to fit current context.

Prior work (Guu et al. 2017) has figured out how to edit prototype in an unconditional setting, but it cannot be applied to the response generation directly. In this paper, we propose a prototype editing method in a conditional setting[1]. Our idea is that differences between responses strongly correlates with differences in their contexts (i.e. if a word in prototype context is changed, its related words in the response are probably modified in the editing.). We realize this idea by designing a context-aware editing model that is built upon a encoder-decoder model augmented with an editing vector. The edit vector is computed by the weighted average of insertion word embeddings and deletion word embeddings. Larger weights mean that the editing model should pay more attention on corresponding words in revision. For instance, in Table 1, we wish words like "dessert", "Tofu" and "vegetables" get larger weights than words like "and" and " at". The encoder learns the prototype representation with a gated recurrent unit (GRU), and feeds the representation to a decoder together with the edit vector. The decoder is a GRU language model, that regards the concatenation of last step word embedding and the edit vector as inputs, and predicts the next word with an attention mechanism.

Our experiments are conducted on a large scale Chinese conversation corpus comprised of 20 million context-response pairs. We compare our model with generative models and retrieval models in terms of fluency, relevance, diversity and originality. The experiments show that our method outperforms traditional generative models on relevance, diversity and originality. We further find that the revised response achieves better relevance compared to its prototype and other retrieval results, demonstrating that the editing process does not only promote response originality but also improve the relevance of retrieval results.

Our contributions are listed as follows: 1) this paper proposes a new paradigm, prototype-then-edit, for response generation; 2) we elaborate a simple but effective context-aware editing model for response generation; 3) we empirically verify the effectiveness of our method in terms of relevance, diversity, fluency and originality.

Related Work

Research on chatbots goes back to the 1960s when ELIZA was designed (Weizenbaum 1966) with a huge amount of hand-crafted templates and rules. Recently, researchers have paid more and more attention on data-driven approaches (Ritter, Cherry, and Dolan 2011; Ji, Lu, and Li 2014) due to their superior scalability. Most of these methods are classified as retrieval-based methods (Ji, Lu, and Li 2014; Yan, Song, and Wu 2016) and generation methods (Li et al. 2016b; 2017; Zhou et al. 2017). The former one aims to select a relevant response using a matching model, while the latter one generates a response with natural language generative models.

Prior works on retrieval-based methods mainly focus on the matching model architecture for single turn conversation (Hu et al. 2014) and multi-turn conversation (Lowe et al. 2015; Zhou et al. 2016; Wu et al. 2017). For the studies of generative methods, a huge amount of work aims to mitigate the "safe response" issue from different perspectives. Most of work build models under a sequence to sequence framework (Sutskever, Vinyals, and Le 2014), and introduce other elements, such as latent variables (Serban et al. 2017), topic information (Xing et al. 2017), and dynamic vocabulary (Wu et al. 2018) to increase response diversity. Furthermore, the reranking technique (Li et al. 2016a), reinforcement learning technique (Li et al. 2016b), and adversarial learning technique (Li et al. 2017; Xu et al. 2017) have also been applied to response generation. Apart from work on "safe response", there is a growing body of literature on style transfer (Fu et al. 2018; Wang et al. 2017) and emotional response generation (Zhou et al. 2017). In general, most of previous work generates a response from scratch either left-to-right or conditioned on a latent vector, whereas our approach aims to generate a response by editing a prototype. Prior works have attempted to utilize prototype responses to guide the generation process (Song et al. 2016; Pandey et al. 2018), in which prototype responses are encoded into vectors and feed to a decoder along with a context representation. Our work differs from previous ones on two aspects. One is they do not consider prototype context in the generation process, while our model utilizes context differences to guide editing process. The other is that we regard prototype responses as a source language, while their works formulate it as a multi-source seq2seq task, in which the current context and prototype responses are all source languages in the generation process.

Recently, some researches have explored natural language generation by editing (Guu et al. 2017; Liao et al. 2018). A typical approach follows a writing-then-edit paradigm, that utilizes one decoder to generate a draft from scratch and uses another decoder to revise the draft (Xia et al. 2017). The other approach follows a retrieval-then-edit paradigm, that uses a Seq2Seq model to edit a prototype retrieved from a corpus (Guu et al. 2017; Li et al. 2018; Cao et al. 2018). As far as we known, we are the first to leverage context lexical differences to edit prototypes.

Background

Before introducing our approach, we first briefly describe state-of-the-art natural language editing method (Guu et al. 2017). Given a sentence pair (x, x'), our goal is to obtain

[1]conditional setting means the editor should consider the context (dialogue history) except in a response itself in the revision. Editor only considers a sentence itself in the unconditional setting

sentence x by editing the prototype x'. The general framework is built upon a Seq2Seq model with an attention mechanism, which takes x' and x as source sequence and target sequence respectively. The main difference is that the generative probability of a vanilla Seq2Seq model is $p(x|x')$ whereas the probability of the edit model is $p(x|x', z)$ where z is an edit vector sampled from a pre-defined distribution like variational auto-encoder. In the training phase, the parameter of the distribution is conditional on the context differences. We first define $I = \{w|w \in x \wedge w \notin x'\}$ as an insertion word set, where w is a word added to the prototype, and $D = \{w'|w' \in x' \wedge w' \notin x\}$ is a deletion word set, where w is a word deleted from the prototype. Subsequently, we compute an insertion vector $\vec{i} = \sum_{w \in I} \Psi(w)$ and a deletion vector $\vec{d} = \sum_{w' \in D} \Psi(w')$ by a summation over word embeddings in two corresponding sets, where $\Psi(\cdot)$ transfers a word to its embedding. Then, the edit vector z is sampled from a distribution whose parameters are governed by the concatenation of $\vec{i}$ and $\vec{d}$. Finally, the edit vector and output of the encoder are fed to the decoder to generate x.

For response generation, which is a conditional setting of text editing, an interesting question raised, that is how to generate the edit by considering contexts. We will introduce our motivation and model in details in the next section.

Approach

Model Overview

Suppose that we have a data set $\mathcal{D} = \{(C_i, R_i)\}_{i=1}^N$. $\forall i$, (C_i, R_i) comprises a context $C_i = (c_{i,1}, \ldots, c_{i,l})$ and its response $R_i = (r_{i,1}, \ldots, r_{i,l})$, where $c_{i,j}$ is the j-th word of the context C_i and $r_{i,j}$ is the j-th word of the response R_i. It should be noted that C_i can be either a single turn input or a multiple turn input. As the first step, we assume C_i is a single turn input in this work, and leave the verification of the same technology for multi-turn response generation to future work. Our full model is shown in Figure 1, consisting of a prototype selector $\mathcal{S}$ and a context-aware neural editor $\mathcal{E}$. Given a new conversational context C, we first use $\mathcal{S}$ to retrieve a context-response pair $(C_i, R_i) \in \mathcal{D}$. Then, the editor $\mathcal{E}$ calculates an edit vector $z_i = f(C_i, C)$ to encode the information about the differences between C_i and C. Finally, we generate a response according to the probability of $p(R|z_i, R_i)$. In the following, we will elaborate how to design the selector $\mathcal{S}$ and the editor $\mathcal{E}$.

Prototype Selector

A good prototype selector $\mathcal{S}$ plays an important role in the prototype-then-edit paradigm. We use different strategies to select prototypes for training and testing. In testing, as we described above, we retrieve a context-response pair (C', R') from a pre-defined index for context C according to the similarity of C and C'. Here, we employ Lucene[2] to construct the index and use its inline algorithm to compute the context similarity.

[2] https://lucenenet.apache.org/

Now we turn to the training phase. $\forall i$, (C_i, R_i), our goal is to maximize the generative probability of R_i by selecting a prototype $(C_i', R_i') \in \mathcal{D}$. As we already know the ground-truth response R_i, we first retrieve thirty prototypes $\{(C_{i,j}', R_{i,j}')\}_{j=1}^{20}$ based on the response similarity instead of context similarity, and then reserve prototypes whose Jaccard similarity to R_i are in the range of $[0.3, 0.7]$. Here, we use Lucene to index all responses, and retrieve the top 20 similar responses along with their corresponding contexts for R_i. The Jaccard similarity measures text similarity from a bag-of-word view, that is formulated as

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}, \tag{1}$$

where A and B are two bags of words and $|\cdot|$ denotes the number of elements in a collection. Each context-response pair is processed with the above procedure, so we obtain enormous quadruples $\{(C_i, R_i, C_{i,j}', R_{i,j}')_{j=0}^{M_i}\}_{i=1}^N$ after this step. The motivation behind filtering out instances with Jaccard similarity < 0.3 is that a neural editor model performs well only if a prototype is lexically similar (Guu et al. 2017) to its ground-truth. Besides, we hope the editor does not copy the prototype so we discard instances where the prototype and groundtruth are nearly identical (i.e. Jaccard similarity > 0.7). We do not use context similarity to construct parallel data for training, because similar contexts may correspond to totally different responses, so-called one-to-many phenomenon (Li et al. 2016a) in dialogue generation, that impedes editor training due to the large lexicon gap. According to our preliminary experiments, the editor always generates non-sense responses if training data is constructed by context similarity.

Context-Aware Neural Editor

A context-aware neural editor aims to revise a prototype to adapt current context. Formally, given a quadruple (C, R, C', R') (we omit subscripts for simplification), a context-aware neural editor first forms an edit vector z using C and C', and then updates parameters of the generative model by maximizing the probability of $p(R|z, R')$. For testing, we directly generate a response after getting the editor vector. In the following, we will introduce how to obtain the edit vector and learn the generative model in details.

Edit Vector Generation For an unconditional sentence editing setting (Guu et al. 2017), an edit vector is randomly sampled from a distribution because how to edit the sentence is not constrained. In contrast, we should take both of C and C' into consideration when we revise a prototype response R'. Formally, R' is firstly transformed to hidden vectors $\{h_k|h_k = \overrightarrow{h}_k \oplus \overleftarrow{h}_k\}_{k=1}^{n_j}$ through a biGRU parameterized as Equation (2).

$$\overrightarrow{h}_j = f_{\text{GRU}}(\overrightarrow{h}_{j-1}, r_j'); \ \overleftarrow{h}_j = f_{\text{GRU}}(\overleftarrow{h}_{j+1}, r_j') \tag{2}$$

where r_j' is the j-th word of R'.

Then we compute a context diff-vector $diff_c$ by an attention mechanism defined as follows

$$diff_c = \sum_{w \in I} \beta_w \Psi(w) \oplus \sum_{w' \in D} \gamma_{w'} \Psi(w'), \tag{3}$$

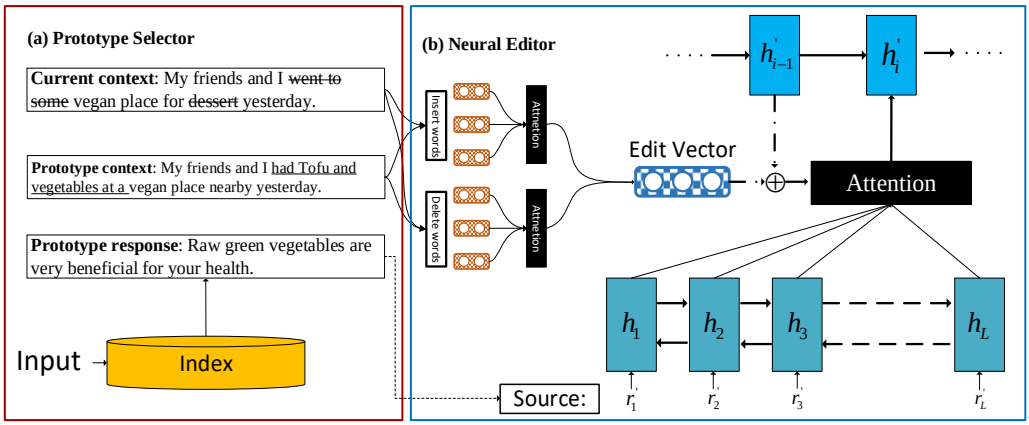

Figure 1: Architecture of our model.

where $\oplus$ is a concatenation operation, $I = \{w | w \in C \wedge w \notin C'\}$ is a insertion word set, and $D = \{w' | w' \in C' \wedge w' \notin C\}$ is a deletion word set. $diff_c$ explicitly encodes insertion words and deletion words from C to C'. β_w is the weight of a insertion word w, that is computed by

$$\beta_w = \frac{exp(e_w)}{\sum_{w \in I} exp(e_w)}, \tag{4}$$

$$e_w = \mathbf{v}_\beta{}^\top tanh(\mathbf{W}_\beta[\Psi(w) \oplus h_l]), \tag{5}$$

where $\mathbf{v}_\beta$ and $\mathbf{W}_\beta$ are parameters, and h_l is the last hidden state of the encoder. $\gamma_{w'}$ is obtained with a similar process:

$$\gamma_{w'} = \frac{exp(e_{w'})}{\sum_{w' \in D} exp(e_{w'})}, \tag{6}$$

$$e_{w'} = \mathbf{v}_\gamma{}^\top tanh(\mathbf{W}_\gamma[\Psi(w') \oplus h_l]), \tag{7}$$

We assume that different words influence the editing process unequally, so we weighted average insertion words and deletion words to form an edit in Equation 3. Table explains our motivation as well, that is "desserts" is much more important than "the" in the editing process. Then we compute the edit vector z by following transformation

$$z = tanh(\mathbf{W} \cdot diff_c + \mathbf{b}), \tag{8}$$

where $\mathbf{W}$ and $\mathbf{b}$ are two parameters. Equation 8 can be regarded as a mapping from context differences to response differences.

It should be noted that there are several alternative approaches to compute $diff_c$ and z for this task, such as applying memory networks, latent variables, and other complex network architectures. Here, we just use a simple method, but it yields interesting results on this task. We will further illustrate our experiment findings in the next section.

Prototype Editing We build our prototype editing model upon a Seq2Seq with an attention mechanism model, which integrates the edit vector into the decoder.

The decoder takes$\{h_k\}_{k=1}^{n_j}$ as an input and generates a response by a GRU language model with attention. The hidden state of the decoder is acquired by

$$h'_j = f_{\text{GRU}}(h'_{j-1}, r_{j-1} \oplus z_i), \tag{9}$$

where the input of j-th time step is the last step hidden state and the concatenation of the $(j-1)$-th word embedding and the edit vector obtained in Equation 8. Then we compute a context vector c_i, which is a linear combination of $\{h_1, \ldots, h_t\}$:

$$c_i = \sum_{j=1}^{t} \alpha_{i,j} h_j, \tag{10}$$

where $\alpha_{i,j}$ is given by

$$\alpha_{i,j} = \frac{exp(e_{i,j})}{\sum_{k=1}^{t} exp(e_{i,k})}, \tag{11}$$

$$e_{i,j} = \mathbf{v}^\top tanh(\mathbf{W}_\alpha[h_j \oplus h'_i]), \tag{12}$$

where $\mathbf{v}$ and $\mathbf{W}_\alpha$ are parameters. The generative probability distribution is given by

$$s(r_i) = softmax(\mathbf{W_p}[r_{i-1} \oplus h'_i \oplus c_i] + \mathbf{b_p}), \tag{13}$$

where $\mathbf{W_p}$ and $\mathbf{b_p}$ are two parameters. Equation 11 and 13 are the attention mechanism (Bahdanau, Cho, and Bengio 2015), that mitigates the long-term dependency issue of the original Seq2Seq model. We append the edit vector to every input embedding of the decoder in Equation 9, so the edit information can be utilized in the entire generation process.

We learn our response generation model by minimizing the negative log likelihood of $\mathcal{D}$

$$\mathcal{L} = -\sum_{i=1}^{N} \sum_{j=1}^{l} \log p(r_{i,j} | z_i, R'_i, r_{i,k<j}). \tag{14}$$

We implement our model by PyTorch [3].We employ the Adam algorithm (Kingma and Ba 2015) to optimize the objective function with a batch size of 128. We set the initial learning rate as 0.001 and reduce it by half if perplexity on validation begins to increase. We will stop training if the perplexity on validation keeps increasing in two successive epochs. .

[3] https://pytorch.org/

Experiment

Experiment setting

In this paper, we only consider single turn response generation. We collected over 20 million human-human context-response pairs (context only contains 1 turn) from Douban Group [4]. After removing duplicated pairs and utterance longer than 30 words, we split 19,623,374 pairs for training, 10,000 pairs for validation and 10,000 pairs for testing. The average length of contexts and responses are 11.64 and 12.33 respectively. The training data mentioned above is used by retrieval models and generative models.

In terms of ensemble models and our editing model, the validation set and the test set are the same with datasets prepared for retrieval and generation models. Besides, for each context in the validation and test sets, we select its prototypes with the method described in Section "Prototype Selector". We follow Song et al. (2016) to construct a training data set for ensemble models, and construct a training data set with the method described in Section "Prototype Selector" for our editing models. We can obtain 42,690,275 (C_i, R_i, C_i', R_i') quadruples with the proposed data preparing method. For a fair comparison, we randomly sample 19,623,374 instances for the training of our method and the ensemble method respectively. To facilitate further research, related resources of the paper can be found at `https://github.com/MarkWuNLP/ResponseEdit`.

Baseline

S2SA: We apply the Seq2Seq with attention (Bahdanau, Cho, and Bengio 2015) as a baseline model. We use a Pytorch implementation, OpenNMT (Klein et al. 2017) in the experiment.

S2SA-MMI: We employed the bidirectional-MMI decoder as in (Li et al. 2016a). The hyperparameter λ is set as 0.5 according to the paper's suggestion. 200 candidates are sampled from beam search for reranking.

CVAE: The conditional variational auto-encoder is a popular method of increasing the diversity of response generation (Zhao, Zhao, and Eskénazi 2017). We use the published code at `https://github.com/snakeztc/NeuralDialog-CVAE`, and conduct small adaptations for our single turn scenario.

Retrieval: We compare our model with two retrieval-based methods to show the effect of editing. One is **Retrieval-default** that directly regards the top-1 result given by Lucene as the reply. The second one is **Retrieval-Rerank**, where we first retrieve 20 response candidates, and then employ a dual-LSTM model (Lowe et al. 2015) to compute matching degree between current context and the candidates. The matching model is implemented with the same setting in (Lowe et al. 2015), and is trained on the training data set where negative instances are randomly sampled with a ratio of $pos : neg = 1 : 9$.

Ensemble Model: Song et al (2016) propose an ensemble of retrieval and generation methods. It encodes current context and retrieved responses (Top-k retrieved responses are all used in the generation process.) into vectors, and feeds these representations to a decoder to generate a new response. As there is no official code, we implement it carefully by ourselves. We use the top-1 response returned by beam search as a baseline, denoted as **Ensemble-default**. For a fair comparison, we further rerank top 20 generated results with the same LSTM based matching model, and denote it as **Ensemble-Rerank**. We further create a candidate pool by merging the retrieval and generation results, and rerank them with the same ranker. The method is denoted as **Ensemble-Merge**.

Correspondingly, we evaluate three variants of our model. Specifically, **Edit-default** and **Edit-1-Rerank** edit top-1 response yielded by Retrieval-default and Retrieval-Rerank respectively. **Edit-N-Rerank** edits all 20 responses returned by Lucene and then reranks the revised results with the dual-LSTM model. We also merge edit results of **Edit-N-Rerank** and candidates returned by the search engine, and then rerank them, which is denoted as **Edit-Merge**. In practice, the word embedding size and editor vector size are 512, and both of the encoder and decoder are a 1-layer GRU whose hidden vector size is 1024. Message and response vocabulary size are 30000, and words not covered by the vocabulary are represented by a placeholder \$UNK\$. Word embedding size, hidden vector size and attention vector size of baselines and our models are the same. All generative models use beam search to yield responses, where the beam size is 20 except S2SA-MMI. For all models, we remove \$UNK\$ from the target vocabulary, because it always leads to a fluency issue in evaluation.

Evaluation Metrics

We evaluate our model on four criteria: fluency, relevance, diversity and originality. We employ Embedding Average (Average), Embedding Extrema (Extrema), and Embedding Greedy (Greedy) (Liu et al. 2016) to evaluate response relevance, which are better correlated with human judgment than BLEU. Following (Li et al. 2016a), we evaluate the response diversity based on the ratios of distinct unigrams and bigrams in generated responses, denoted as Distinct-1 and Distinct-2. In this paper, we define a new metric, originality, that is defined as the ratio of generated responses that do not appear in the training set. Here, "appear" means we can find exactly the same response in our training data set. We randomly select 1,000 contexts from the test set, and ask three native speakers to annotate response fluency. We conduct 3-scale rating: +2, +1 and 0. +2: The response is fluent and grammatically correct. +1: There are a few grammatical errors in the response but readers could understand it. 0: The response is totally grammatically broken, making it difficult to understand. As how to evaluate response generation automatically is still an open problem (Liu et al. 2016), we further conduct human evaluations to compare our models with baselines. We ask the same three native speakers to do a side-by-side comparison (Li et al. 2016b) on the 1,000 contexts. Given a context and two responses generated by different models, we ask annotators to decide which response is better (Ties are permitted).

[4] `https://www.douban.com/group` Douban is a popular forum in China.

Table 2: Automatic evaluation results. Numbers in bold mean that improvement from the model on that metric is statistically significant over the baseline methods (t-test, p-value < 0.01). κ denotes Fleiss Kappa (Fleiss 1971), which reflects the agreement among human annotators.

	Relevance			**Diversity**		**Originality**	**Fluency**				
	Average	Extrema	Greedy	Distinct-1	Distinct-2	Not appear	+2	+1	0	Avg	κ
S2SA	0.346	0.180	0.350	0.032	0.087	0.208	94.0%	5.2%	0.8%	**1.932**	0.89
S2SA-MMI	0.379	0.189	0.385	0.039	0.127	0.297	91.6%	7.6%	0.8%	1.908	0.83
CVAE	0.360	0.183	0.363	0.062	0.178	0.745	83.8%	12.7%	3.5%	1.803	0.84
Retrieval-default	0.288	0.130	0.309	0.098	**0.589**	0.000	92.8%	6.8%	0.4%	1.924	0.88
Retrieval-Rerank	0.380	0.191	0.381	**0.106**	0.560	0.000	91.7%	7.8%	0.5%	1.912	0.88
Ensemble-default	0.352	0.183	0.362	0.035	0.097	0.223	91.3%	7.2%	1.5%	1.898	0.84
Ensemble-Rerank	0.372	0.187	0.379	0.040	0.135	0.275	90.4%	7.8%	1.8%	1.898	0.84
Edit-default	0.297	0.150	0.327	0.071	0.300	0.796	89.6%	9.2%	1.2%	1.884	0.87
Edit-1-Rerank	0.367	0.185	0.371	0.077	0.296	0.794	93.2%	5.6%	1.2%	1.920	0.79
Edit-N-Rerank	**0.386**	**0.203**	**0.389**	0.068	0.280	**0.860**	87.2%	10.8%	2.0%	1.852	0.85
Ensemble-Merge	0.385	0.193	0.387	0.058	0.247	0.134	91.2%	7.3%	1.5%	1.897	0.89
Edit-Merge	0.391	0.197	0.392	0.103	0.443	0.734	88.6%	9.7%	1.7%	1.869	0.85

Table 3: Human side-by-side evaluation results. If a row is labeled as "a v.s.b", the second column, "loss", means the ratio of responses given by "a" are worse than those given by"b".

	Loss	Tie	Win	κ
Ed-Default v.s. R-Default	23.6	41.2	35.2	0.54
Ed-Default v.s. Ens-Default	29.4	47.8	22.8	0.59
Ed-1-Rerank v.s. R-Rerank	29.2	45.3	25.5	0.60
Ed-N-Rerank v.s. Ens-Rerank	24.9	42.1	33.0	0.55
Ed-N-Rerank v.s. R-Rerank	25.2	44.8	30.0	0.62

Evaluation Results

Table 2 shows the evaluation results on the Chinese dataset. Our methods are better than retrieval-based methods on embedding based metrics, that means revised responses are more relevant to ground-truth in the semantic space. Our model just slightly revises prototype response, so improvements on automatic metrics are not that large but significant on statistical tests (t-test, p-value < 0.01). Two factors are known to cause Edit-1-Rerank worse than Retrieval-Rerank. 1) Rerank algorithm is biased to long responses, that poses a challenge for the editing model. 2) Despite of better prototype responses, a context of top-1 response is always greatly different from current context, leading to a large insertion word set and a large deletion set, that also obstructs the revision process. In terms of diversity, our methods drop on distinct-1 and distinct-2 in a comparison with retrieval-based methods, because the editing model often deletes special words pursuing for better relevance. Retrieval-Rerank is better than retrieval-default, indicating that it is necessary to rerank responses by measuring context-response similarity with a matching model.

Our methods significantly outperform generative baselines in terms of diversity since prototype responses are good start-points that are diverse and informative. It demonstrates that the prototype-then-editing paradigm is capable of addressing the safe response problem. Edit-Rerank is better than generative baselines on relevance but Edit-default is not, indicating a good prototype selector is quite important to our editing model. In terms of originality, about 86% revised response do not appear in the training set, that surpasses S2SA, S2SA-MMI and CVAE. This is mainly because baseline methods are more likely to generate safe responses that are frequently appeared in the training data, while our model tends to modify an existing response that avoids duplication issue. In terms of fluency, S2SA achieves the best results, and retrieval based approaches come to the second place. Safe response enjoys high score on fluency, that is why S2SA and S2SA-MMI perform well on this metric. Although editing based methods are not the best on the fluency metric, they also achieve a high absolute number. That is an acceptable fluency score for a dialogue engine, indicating that most of generation responses are grammatically correct. In addition, in terms of the fluency metric, Fleiss' Kappa (Fleiss 1971) on all models are around 0.8, showing a high agreement among labelers.

Compared to ensemble models, our model performs much better on diversity and originality, that is because we regard prototype response instead of the current context as source sentence in the Seq2Seq, which keeps most of content in prototype but slightly revises it based on the context difference. Both of the ensemble and edit model are improved when the original retrieval candidates are considered in the rerank process.

Regarding human side-by-side evaluation, we can find that Edit-Default and Edit-N-rerank are slightly better than Retrieval-default and Retrieval-rerank (The winning examples are more than the losing examples), indicating that the post-editing is able to improve the response quality. Ed-Default is worse than Ens-Default, but Ed-N-Rerank is better than Ens-Rerank. This is mainly because the editing model regards the prototype response as the source language, so it is highly depends on the quality of prototype response.

Discussions

Model ablation We train variants of our model by removing the insertion word vector, the deletion word vector, and both of them respectively. The results are shown in Table 4. We can find that embedding based metrics drop dramati-

Table 4: Model ablation tests. Full model denotes Edit-N-Rerank. "-del" means we only consider insertion words. "-ins" means we only consider deletion words. "-both" means we train a standard seq2seq model from prototype response to revised response.

	Average	Extrema	Greedy	Dis-1	Dis-2	Originality
Full	38.6	20.3	38.9	0.068	0.280	86.0
- ins	34.2	17.9	34.8	0.067	0.286	85.1
- del	34.0	17.6	34.6	0.065	0.278	86.3
-both	32.8	16.9	33.7	0.067	0.282	85.4

Table 5: Case Study. We show examples yielded by Edit-default and Edit-Rerank. Chinese utterances are translated to English here.

	Insertion case
Context	身 在 国外 寂寞 无聊 就 化妆 When I feel bored and lonely abroad, I like makeup.
Prototype context	无聊 就 玩 If I feel bored, I will play something.
Prototype response	嗯 OK.
Revised response	嗯 我 也 喜欢 化妆 OK. I love makeup too.
	Deletion case
Context	我 比较black 常 吃 辛拉面 I often eat spice noodles.
Prototype context	我 在 台湾 时候 常 吃 辛拉面 和 卤肉饭。 When I lived in Taiwan, I often eat spicy Noodles and braised pork rice.
Prototype response	我 也 喜欢 卤肉饭。 I love braised pork rice as well.
Revised response	我 也 喜欢。 I love it as well. (In Chinese, model just deletes the phrase "braised pork rice" without adding any word.)
	Replacement case
Context	纹身 有没有 办法 全部 弄 干净 Is there any way to get all tattoos clean
Prototype context	把 药 抹 头发 上 能 把 头发 弄 干净 么 Is it possible to clean your hair by wiping the medicine on your hair?
Prototype response	抹 完 真的 头发 干净 很多 After wiping it, hair gets clean much.
Revised response	抹完 纹身 就 会 掉 很多 After wiping it, most of tattoos will be cleaned.

cally when the editing vector is partially or totally removed, indicating that the edit vector is crucial for response relevance. Diversity and originality do not decrease after the edit vector is removed, implying that the retrieved prototype is the key factor for these two metrics. According to above observations, we conclude that the prototype selector and the context-aware editor play different roles in generating responses.

Editing Type Analysis It is interesting to explore the semantic gap between prototype and revised response. We ask annotators to conduct 4-scale rating on 500 randomly sampled prototype-response pairs given by Edit-default and Edit-N-Rerank respectively. The 4-scale is defined as: identical, paraphrase, on the same topic and unrelated.

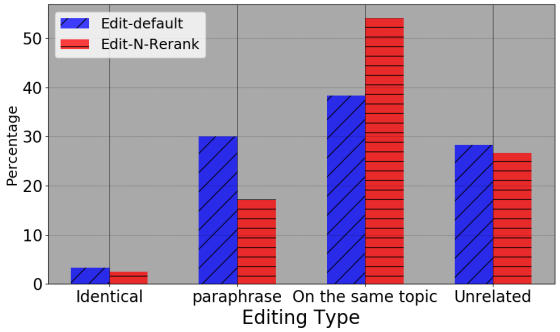

Figure 2: Distribution across different editing types.

Figure 2 provides the ratio of four editing types defined above. For both methods, Only 2% of edits are exactly the same with the prototype, that means our model does not downgrade to a copy model. Surprisingly, there are 30% revised responses are unrelated to prototypes. The key factor for this phenomenon is that the neural editor will rewrite the prototype when it is hard to insert insertion words to the prototype. The ratio of "on the same topic" response given by Edit-N-rerank is larger than Edit-default, revealing that "on the same topic" responses might be more relevant from the view of a LSTM based reranker.

Case Study We give three examples to show how our model works in Table 5. The first case illustrates the effect of word insertion. Our editing model enriches a short response by inserting words from context, that makes the conversation informative and coherent. The second case gives an example of word deletion, where a phrase "braised pork rice" is removed as it does not fit current context. Phrase "braised pork rice" only appears in the prototype context but not in current context, so it is in the deletion word set D, that makes the decoder not generate it. The third one is that our model forms a relevant query by deleting some words in the prototype while inserting other words to it. Current context is talking about "clean tatoo", but the prototype discusses "clean hair", leading to an irrelevant response. After the word substitution, the revised response becomes appropriated for current context.

Editing Words According to our observation, function words and nouns are more likely to be added/deleted. This

is mainly because function words, such as pronoun, auxiliary, and interjection may be substituted in the paraphrasing. In addition, a large proportion of context differences is caused by nouns substitutions, thus we observe that nouns are added/deleted in the revision frequently.

Conclusion

We present a new paradigm, prototype-then-edit, for open domain response generation, that enables a generation-based chatbot to leverage retrieved results. We propose a simple but effective model to edit context-aware responses by taking context differences into consideration. Experiment results on a large-scale dataset show that our model outperforms traditional methods on some metrics. In the future, we will investigate how to jointly learn the prototype selector and neural editor.

Acknowledgments

Yu is supported by AdeptMind Scholarship and Microsoft Scholarship. This work was supported in part by the Natural Science Foundation of China (Grand Nos. U1636211, 61672081, 61370126), Beijing Advanced Innovation Center for Imaging Technology (No.BAICIT-2016001) and National Key R&D Program of China (No.2016QY04W0802).

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. *ICLR*.

Cao, S. Z.; Li, W.; Wei, F.; and Li, S. 2018. Retrieve, rerank and rewrite: Soft template based neural summarization. In *ACL*, volume 2, 3.

Fleiss, J. L. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin* 76(5):378.

Fu, Z.; Tan, X.; Peng, N.; Zhao, D.; and Yan, R. 2018. Style transfer in text: Exploration and evaluation. In *AAAI 2018*.

Guu, K.; Hashimoto, T. B.; Oren, Y.; and Liang, P. 2017. Generating sentences by editing prototypes. *CoRR* abs/1709.08878.

Hu, B.; Lu, Z.; Li, H.; and Chen, Q. 2014. Convolutional neural network architectures for matching natural language sentences. In *NIPS*, 2042–2050.

Ji, Z.; Lu, Z.; and Li, H. 2014. An information retrieval approach to short text conversation. *arXiv preprint arXiv:1408.6988*.

Kingma, D., and Ba, J. 2015. Adam: A method for stochastic optimization. *ICLR*.

Klein, G.; Kim, Y.; Deng, Y.; Senellart, J.; and Rush, A. M. 2017. Opennmt: Open-source toolkit for neural machine translation. In *Proc. ACL*.

Li, J.; Galley, M.; Brockett, C.; Gao, J.; and Dolan, B. 2016a. A diversity-promoting objective function for neural conversation models. In *NAACL, 2016*, 110–119.

Li, J.; Monroe, W.; Ritter, A.; Jurafsky, D.; Galley, M.; and Gao, J. 2016b. Deep reinforcement learning for dialogue generation. In *EMNLP 2016*, 1192–1202.

Li, J.; Monroe, W.; Shi, T.; Ritter, A.; and Jurafsky, D. 2017. Adversarial learning for neural dialogue generation. *EMNLP*.

Li, J.; Jia, R.; He, H.; and Liang, P. 2018. Delete, retrieve, generate: A simple approach to sentiment and style transfer. *NAACL*.

Liao, Y.; Bing, L.; Li, P.; Shi, S.; Lam, W.; and Zhang, T. 2018. Incorporating pseudo-parallel data for quantifiable sequence editing. *arXiv preprint arXiv:1804.07007*.

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. *EMNLP*.

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. *SIGDIAL*.

Pandey, G.; Contractor, D.; Kumar, V.; and Joshi, S. 2018. Exemplar encoder-decoder for neural conversation generation. In *ACL*, volume 1, 1329–1338.

Ritter, A.; Cherry, C.; and Dolan, W. B. 2011. Data-driven response generation in social media. In *EMNLP*, 583–593.

Serban, I. V.; Sordoni, A.; Bengio, Y.; Courville, A. C.; and Pineau, J. 2016. End-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, 3776–3784.

Serban, I. V.; Sordoni, A.; Lowe, R.; Charlin, L.; Pineau, J.; Courville, A. C.; and Bengio, Y. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*, 3295–3301.

Shang, L.; Lu, Z.; and Li, H. 2015. Neural responding machine for short-text conversation. In *ACL 2015*.

Song, Y.; Yan, R.; Li, X.; Zhao, D.; and Zhang, M. 2016. Two are better than one: An ensemble of retrieval- and generation-based dialog systems. *arXiv preprint arXiv:1610.07149*.

Sordoni, A.; Galley, M.; Auli, M.; Brockett, C.; Ji, Y.; Mitchell, M.; Nie, J.; Gao, J.; and Dolan, B. 2015. A neural network approach to context-sensitive generation of conversational responses. In *NAACL 2015*, 196–205.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, 3104–3112.

Vinyals, O., and Le, Q. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Wang, D.; Jojic, N.; Brockett, C.; and Nyberg, E. 2017. Steering output style and topic in neural response generation. In *EMNLP 2017*, 2140–2150.

Weizenbaum, J. 1966. Eliza: a computer program for the study of natural language communication between man and machine. *Communications of the ACM* 9(1):36–45.

Wu, Y.; Wu, W.; Xing, C.; Zhou, M.; and Li, Z. 2017. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. In *ACL 2017*, 496–505.

Wu, Y.; Wu, W.; Yang, D.; Xu, C.; and Li, Z. 2018. Neural response generation with dynamic vocabularies. In *AAAI 2018*.

Xia, Y.; Tian, F.; Wu, L.; Lin, J.; Qin, T.; Yu, N.; and Liu, T. 2017. Deliberation networks: Sequence generation beyond one-pass decoding. In *NIPS 2017*, 1782–1792.

Xing, C.; Wu, W.; Wu, Y.; Liu, J.; Huang, Y.; Zhou, M.; and Ma, W.-Y. 2017. Topic aware neural response generation. In *AAAI*, volume 17, 3351–3357.

Xu, Z.; Liu, B.; Wang, B.; Sun, C.; Wang, X.; Wang, Z.; and Qi, C. 2017. Neural response generation via GAN with an approximate embedding layer. In *EMNLP 2017*, 617–626.

Yan, R.; Song, Y.; and Wu, H. 2016. Learning to respond with deep neural networks for retrieval-based human-computer conversation system. In *SIGIR 2016*, 55–64.

Zhao, T.; Zhao, R.; and Eskénazi, M. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *ACL 2017*, 654–664.

Zhou, X.; Dong, D.; Wu, H.; Zhao, S.; Yu, D.; Tian, H.; Liu, X.; and Yan, R. 2016. Multi-view response selection for human-computer conversation. In *EMNLP 2016*, 372–381.

Zhou, H.; Huang, M.; Zhang, T.; Zhu, X.; and Liu, B. 2017. Emotional chatting machine: Emotional conversation generation with internal and external memory. *arXiv preprint arXiv:1704.01074*.