Skeleton-to-Response: Dialogue Generation Guided by Retrieval Memory

Deng Cai[*]
The Chinese University of Hong Kong
thisisjcykcd@gmail.com

Yan Wang
Tencent AI Lab
brandenwang@tencent.com

Wei Bi
Tencent AI Lab
victoriabi@tencent.com

Zhaopeng Tu
Tencent AI Lab
zptu@tencent.com

Xiaojiang Liu
Tencent AI Lab
kieranliu@tencent.com

Wai Lam
The Chinese University of Hong Kong
wlam@se.cuhk.edu.hk

Shuming Shi
Tencent AI Lab
shumingshi@tencent.com

Abstract

For dialogue response generation, traditional generative models generate responses solely from input queries. Such models rely on insufficient information for generating a specific response since a certain query could be answered in multiple ways. Consequentially, those models tend to output generic and dull responses, impeding the generation of informative utterances. Recently, researchers have attempted to fill the information gap by exploiting information retrieval techniques. When generating a response for a current query, similar dialogues retrieved from the entire training data are considered as an additional knowledge source. While this may harvest massive information, the generative models could be overwhelmed, leading to undesirable performance. In this paper, we propose a new framework which exploits retrieval results via a skeleton-then-response paradigm. At first, a skeleton is generated by revising the retrieved responses. Then, a novel generative model uses both the generated skeleton and the original query for response generation. Experimental results show that our approaches significantly improve the diversity and informativeness of the generated responses.

Introduction

This paper focuses on tackling the challenges to develop a chit-chat style dialogue system (also known as chatbot). Chit-chat style dialogue system aims at giving meaningful and coherent responses given a dialogue query in open domain. Most modern chit-chat systems can be categorized into two categories, namely, information retrieval-based (IR) models and generative models.

The IR-based models (Ji, Lu, and Li 2014; Hu et al. 2014) directly copy an existing response from a training corpus when receiving a response request. Since the training corpus is usually collected from real-world conversations and possibly post-edited by human, the retrieved responses are informative and grammatical. However, the performance of such systems drops when a given dialogue history is essentially different from those in the training corpus.

The generative models (Shang, Lu, and Li 2015; Vinyals and Le 2015; Li et al. 2016a), on the other hand, generate a new utterance from scratch. While those generative models have better generalization capacity in rare or unseen dialogue contexts, the generated responses tend to be universal

---

and non-informative (e.g., "I don't know", "I think so" etc) rather than meaningful and specific (Li et al. 2016a). It is partly due to the diversity of possible responses to a single query (i.e., the *one-to-many* problem). The dialogue query alone cannot decide a meaningful and specific response. Thus a well-trained model tends to generate the most frequent (safe) responses as reflected in the training corpus.

To summarize, IR-based models may give informative but inappropriate responses while generative models often do the opposite. Given that each methodology has its merits, it is desirable to combine them together. Song et al. (2016) used an extra encoder to transform the retrieved response into dense representation. The resulted representation, together with the representation of the original query, is used to feed the decoder in a standard SEQ2SEQ model. Yet Weston, Dinan, and Miller (2018) used a single encoder that takes the concatenation of the original query and the retrieved as input. Wu et al. (2018) noted that the retrieved information should be used in awareness of the context difference, and further proposed to construct an edit vector by explicitly encoding the lexical differences between the current query and the retrieved query.

However, in our preliminary experiments, we found that the IR-guided models are inclined to degenerate into a copy mechanism, in which the generative models simply repeat the retrieved response without necessary modifications. Drastic performance drop is caused when the retrieved response is irrelevant to the current query. A possible reason is that these methods attempt to implicitly separate the useful information from the other semantics of the retrieved responses in dense vector representations, where all information is mixed together in an uninterpretable way.

To address the above issue, we propose a new framework, skeleton-then-response, for response generation. Our motivations are two-folds: **(1)** The guidance from IR results should only specify a response aspect or pattern, but leave the query-specific details to be elaborated by the generative model itself; **(2)** The retrieval results typically contain excessive information, including some inappropriate or misleading words. It is necessary to filter out irrelevant words and derive a useful skeleton before use.

Our approach consists of two components: a skeleton generator and a response generator. The skeleton generator extracts a response skeleton by detecting and removing un-

wanted words. The response generator is responsible for adding query-specific details to the generated skeleton for query-to-response generation. A dialogue example illustrating our idea is shown in Fig. 1. Because of the discrete choice of skeleton words, the gradient in the training process is no longer differentiable from the response to the skeleton generator. Two techniques are proposed to solve this issue. The first technique is to employ the policy gradient method for rewarding the output of the skeleton generator based on the feedback from a pre-trained critic. An alternative technique is to solve both the skeleton generation and the response generation in a multi-task learning fashion.

Our contributions are summarized as below: (1) We develop a novel framework to inject the power of IR results into generative response models by introducing the idea of skeleton generation; (2) Our approach generates response skeletons by detecting and removing unnecessary words, which facilitates the generation of specific responses while not spoiling the generalization ability of the underlying generative models; (3) Experimental results show that our approach significantly outperforms other compared methods, resulting in more informative and specific responses.

Models

Overview

In this work, we propose to construct a response skeleton based on the result of IR systems for guiding the response generation. The skeleton-then-response paradigm helps reduce the output space of possible responses and provides useful elements missing in the current query.

For each query $q \in Q$, a set of historical query-response pairs $R_q = \{(q'_i, r'_i)\}_{i=1}^N$ are retrieved by some IR techniques. We estimate the generation probability of a response r conditioned on q and R_q. The whole process is decomposed into two parts. First, we assume that there exists a probabilistic model $P_{\theta_{ske}}(t_i|q, q'_i, r'_i)$ mapping each (q, q'_i, r'_i) to a response skeleton t_i. Basically, we mask some parts (ideally useless or unnecessary parts) of a retrieved response for producing a response skeleton. Armed with this skeleton, the final response is generated by revising the skeletons $T = \{t_i\}_{i=1}^N$ by $P_{\theta_{res}}(r|q, T)$. Our overall model consists of two components, namely, the skeleton generator and the response generator. These components are parameterized by the above two probabilistic models, denoted by θ_{ske} and θ_{res} respectively.

For clarity, the proposed model is explained in detail under the default setting of $N = 1$ (i.e., $R_q = (q', r')$) in the following part of this section. It should be noted that our model is readily extended to incorporate multiple IR results. Fig. 2 depicts the architecture of our proposed framework.

Skeleton Generator

The skeleton generator transforms a retrieved response into a skeleton by explicitly removing inappropriate or useless information regarding the current query q. We consider this procedure as a series of word-level masking actions. Following (Wu et al. 2018), we first construct an edit vector by comparing the difference between the original query q and
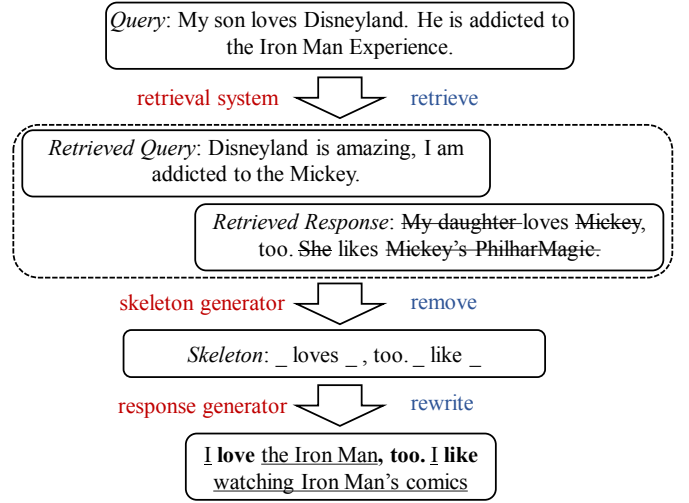

Figure 1: Our idea of leveraging the retrieved query-response pair. It first constructs a response skeleton by removing some words in the retrieved response, then a response is generated via rewriting based on the skeleton.

the retrieved query q'. In (Wu et al. 2018) the edit vector is used to guide the response generation directly. In our model, the edit vector is used to estimate the probability of being reserved or being masked for every word in a sentence. We define two word sets, namely *insertion words I* and *deletion words D*. The insertion words include words that are in the original query q, but not in the retrieved query q', while the deletion words do the opposite.

The two bags of words highlight the changes in the dialogue context, corresponding to the changes in the response. The edit vector z is thus defined as the concatenation of the representations of the two bags of words. We use the weighted sum of the word embeddings to get the dense representations of I and D. The edit vector is computed as:

$$z = \sum_{w_1 \in I} \alpha_{w_1} \Phi(w_1) \oplus \sum_{w_2 \in D} \beta_{w_2} \Phi(w_2), \quad (1)$$

where $\oplus$ is the concatenation operation. Φ maps a word to its corresponding embedding vector, α_{w_1} and β_{w_2} are the weights of an insertion word w_1 and a deletion word w_2 respectively. The weights of different words are derived by an attention mechanism (Luong, Pham, and Manning 2015). Formally, $r' = (r'_1, r'_2 \ldots, r'_{|r'|})$ is processed by a bidirectional GRU network (biGRU). We denote the states of the biGRU (i.e. concatenation of forward and backward GRU states) as $(h_1, h_2, \ldots, h_{|r'|})$. The weight α_{w_1} is calculated by:

$$\alpha_{w_1} = \frac{\exp(s_{w_1})}{\sum_{w \in I} \exp(s_w)},$$
$$s_{w_1} = v_I^\top \tanh(W_I[\Phi(w_1) \oplus h_{|r'|}]), \quad (2)$$

where v_I and W_I are learnable parameters. The weight β_{w_2} is obtained in a similar way with another set of parameters v_D and W_D.
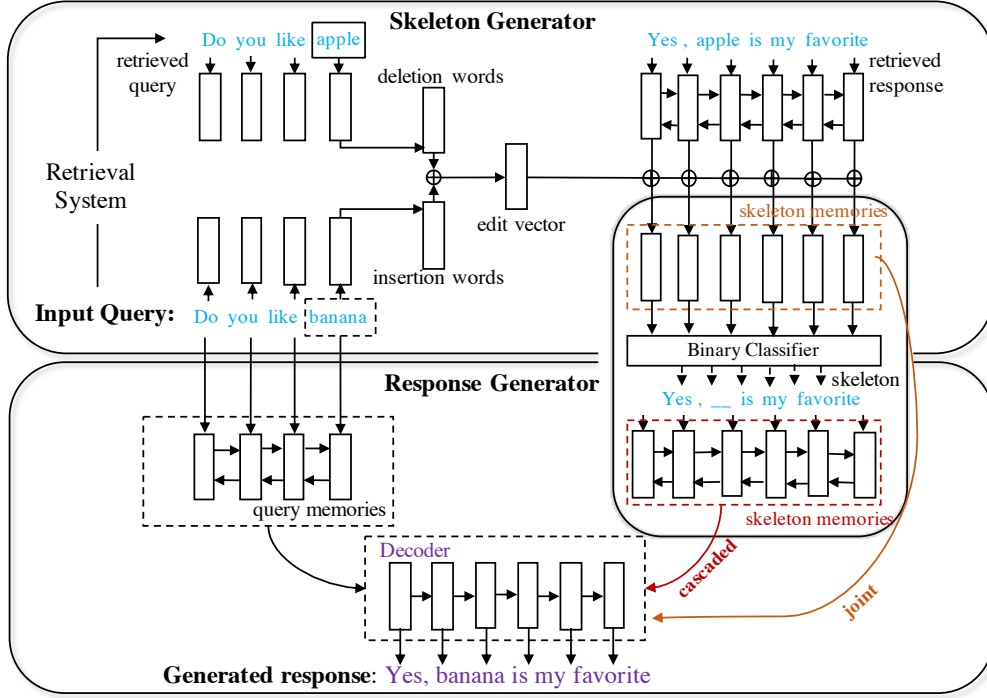
Figure 2: The architecture of our framework. Given a query "Do you like banana", a similar historical query "Do you like apple" is retrieved along with its response, i.e., "Yes, apple is my favorite". Upper: The skeleton generator removes inappropriate words and extracts a response skeleton. Lower: The response generator generates a response based on both the skeleton and the query.

After acquiring the edit vector, we transform the prototype response r' to a skeleton t by the following equations:

$$t = (\phi(r'_1, h_1, z), \phi(r'_2, h_2, z), \cdots, \phi(r'_{|r'|}, h_{|r'|}, z)),$$

$$\phi(r'_i, h_i, z) = \begin{cases} < \text{blank} > & \text{if } \hat{m}_i = 0, \\ r'_i & \text{else} \end{cases}, \qquad (3)$$

where $\hat{m}_i$ is the indicator and equals 0 if r'_i is replaced with a placeholder "<blank>" and 1 otherwise. The probability of $\hat{m}_i = 1$ is computed by

$$P(\hat{m}_i = 1) = \text{sigmoid}(W_m[h_i \oplus z] + b_m). \qquad (4)$$

Response Generator

The response generator can be implemented using most existing IR-augmented models (Song et al. 2016; Weston, Dinan, and Miller 2018; Pandey et al. 2018), just by replacing the retrieved response input with the corresponding skeleton. We discuss our choices below.

Encoders   Two separate bidirectional LSTM (biLSTM) networks are used to obtain the distributed representations of the query memories and the skeleton memories, respectively. For biLSTM, the concatenation of the forward and the backward hidden states at each token position is considered a memory slot, producing two memory pools:

$\mathcal{M}_q = \{h_1, h_2, \ldots, h_{|q|}\}$ for the input query, and $\mathcal{M}_t = \{h'_1, h'_2, \ldots, h'_{|t|}\}$ for the skeleton.[1]

Decoder   During the generation process, our decoder reads information from both the query and the skeleton using attention mechanism (Bahdanau, Cho, and Bengio 2014; Luong, Pham, and Manning 2015). To query the memory pools, the decoder uses the hidden state s_t of itself as the searching key. The matching score function is implemented by bilinear functions:

$$\alpha(h_k, s_t) = h_k{}^T W_q s_t, \quad \beta(h'_k, s_t) = h'_k{}^T W_t s_t, \qquad (5)$$

where W_q and W_t are trainable parameters. A query context vector c_t is then computed as a weighted sum of all memory slots in $\mathcal{M}_q$, where the weight for a memory slot h_k is $\exp(\alpha(h_k, s_t))/(\sum_{i=1}^{|q|} \exp(\alpha(h_i, s_t)))$. A skeleton context vector c'_t is computed in a similar spirit by using $\beta(h'_k, s_t)$'s. The probability of generating the next word r_t is then jointly determined by the decoder's state s_t, the query context c_t and the skeleton context c'_t. We first fuse the information of s_t and c_t by a linear transformation. For c'_t, a gating mechanism is additionally introduced to control the information flow from skeleton memories. Formally, the probability of the next token r_t is estimated by y_t followed by a softmax

_____

[1]Note the skeleton memory pool $\mathcal{M}_t$ could contain multiple response skeletons, further discussed in the experiment section.

Algorithm 1 Proxy Skeleton Construction

Input: a training quadruplet (q, q', r, r'), stop word list S
Output: the proxy skeleton t, the proxy labels m.
1: $r^*, r'^* \leftarrow$ remove the stop words in r and r'
2: $x \leftarrow \text{LongestCommonSubsequence}(r^*, r'^*)$
3: **for** $i = 1$ to $|r'|$ **do**
4:      $m_i \leftarrow 1$ if $(r'_i \in x$ and $r'_i \notin S)$ else 0
5:      $t_i \leftarrow r'_i$ if $(m_i = 1)$ else "<blank>"
6: **end for**
7: **return** t, m

---

function over the vocabulary:

$$y_t = (W_c[s_t \oplus c_t]) \cdot g_t + c'_t \cdot (1 - g_t), \qquad (6)$$

where $g_t = f_g(s_t, c_t, c'_t)$ is implemented by a single layer neural network with sigmoid output layer.

Learning

Given that our skeleton generator performs a non-differentiable hard masking, the overall model cannot be trained end-to-end using the standard maximum likelihood estimate (MLE). A possible solution that circumvents this problem is to treat the skeleton generation and the response generation as two parallel tasks, and solve them jointly in a multi-task learning fashion. An alternative is to bridge the skeleton generator and the final response output using reinforcement learning (RL) methods, which can exclusively inform the skeleton generator with the ultimate goal. The latter option is referred as *cascaded integration* while the former is called *joint integration*.

Recall that we have formulated the skeleton generation as a series of binary classifications. Nevertheless, most of the dialogue datasets are end-to-end query-response pairs without explicit skeletons. Hence, we propose to construct *proxy skeleton*s to facilitate the training.

Definition 1 Proxy Skeleton: *Given a training quadruplet (q, q', r, r') and a stop word list S, the proxy skeleton for r is generated by replacing some tokens in r' with a placeholder "<blank>". A token r'_i is kept if and only if it meets the following conditions*
   1. $r'_i \notin S$
   2. r'_i is a part of the longest common sub-sequence (LCS) (Wagner and Fischer 1974) of r and r'.

The detailed construction process is given in Algorithm 1. The proxy skeletons are used in different manners according to the integration method, which we will introduce below.

Joint Integration   To avoid breaking the differentiable computation, we connect the skeleton generator and the response generator via shared network architectures rather than by passing the discrete skeletons. Concretely, the last hidden states in our skeleton generator (i.e, the hidden states that are utilized to make the masking decisions) are directly used as the skeleton memories in response generation. The skeleton generation and response generation are considered as two tasks. For skeleton generation, the object is to maxi-

mize the log likelihood of the proxy skeleton labels:

$$L(\theta_{ske}) = \sum_{i=1}^{|r'|} \log P(m_i | q, q', r'), \qquad (7)$$

while for response generation, it is trained to maximize the following log likelihood:

$$L(\theta_{res}) = \sum_{i=1}^{|r|} \log P(r_i | r_{1:i-1}, q, t). \qquad (8)$$

The joint network is then trained to maximize two parts of log likelihood:

$$L(\theta_{res} \cup \theta_{ske}) = L(\theta_{res}) + \eta L(\theta_{ske}), \qquad (9)$$

where η is a harmonic weight, and it is set as 1.0 in our experiments.

Cascaded Integration   Now we start to describe how RL methods can be applied to optimize the full model while keeping it running as cascaded process. We regard the skeleton generator as the first RL agent, and the response generator as the second one. The final output generated by the pipeline process and the intermediate skeleton are denoted by $\hat{r}$ and $\hat{t}$ respectively. Given the original query q and the generated response $\hat{r}$, a reward $R(q, \hat{r})$ for generating $\hat{r}$ is calculated. All network parameters are then optimized to maximize the expected reward by the policy gradient. According to the policy gradient theorem (Williams 1992), the gradient for the first agent is

$$\nabla_{\theta_{ske}} J(\theta_{ske}) = \mathbb{E}[R \cdot \nabla \log(P(\hat{t}|q, q', r'))], \qquad (10)$$

and the gradient for the second agent is

$$\nabla_{\theta_{res}} J(\theta_{res}) = \mathbb{E}[R \cdot \nabla \log(P(\hat{r}|q, \hat{t}))]. \qquad (11)$$

The reward function R should convey both the naturalness of the generated response and its relevance to the given query q. A pre-trained critic is utilized to make the judgment. Inspired by comparative adversarial learning in (Li et al. 2018), we design the critic as a classifier that receives four inputs every time: the query q, a human-written response r, a machine-generated response $\hat{r}$ and a random response $\overline{r}$ (yet written by human). The critic is trained to correctly pick the human-written response r among others. Formally, the following objective is maximized:

$$\log D(r|q, \hat{r}, \overline{r}, r) = \log \frac{\exp(h_r{}^{\mathsf{T}} M_D h_q)}{\sum_{x \in \{\hat{r}, \overline{r}, r\}} \exp(h_x{}^{\mathsf{T}} M_D h_q)}, \tag{12}$$

where h_x is a vector representation of x, produced by a bidirectional LSTM (the last hidden state), and M_D is a trainable matrix.[2] The reward function of $\hat{r}$ is defined as:

$$R(q, \hat{r}) = \log \frac{\exp(h_{\hat{r}}{}^{\mathsf{T}} M_D h_q)}{\sum_{x \in \{\hat{r}, \overline{r}, r\}} \exp(h_x{}^{\mathsf{T}} M_D h_q)}. \tag{13}$$

However, when randomly initialized, the skeleton generator and the response generator transmit noisy signals to each other, which leads to sub-optimal policies. We hence propose pre-training each component using Equation (7) and (8) sequentially.

---

[2]Note the classifier could be fine-tuned with the training of our generators, which falls into the adversarial learning setting.

Related Work

Multi-source Dialogue Generation Chit-chat style dialogue system dates back to ELIZA (Weizenbaum 1966). Early work uses handcrafted rules, while modern systems usually use data-driven approaches, e.g., information retrieval techniques. Recently, end-to-end neural approaches (Vinyals and Le 2015; Serban et al. 2016; Li et al. 2016a; Sordoni et al. 2015) have attracted increasing interest. For those generative models, a notorious problem is the "safe response" problem: the generated responses are dull and generic, which may attribute to the lack of sufficient input information. The query alone cannot specify an informative response. To mitigate the issue, many research efforts have been paid to introducing other information source, such as unsupervised latent variable (Serban et al. 2017; Zhao, Lee, and Eskenazi 2018; Cao and Clark 2017; Shen et al. 2017), discourse-level variations (Zhao, Zhao, and Eskenazi 2017), topic information (Xing et al. 2017), speaker personality (Li et al. 2016b) and knowledge base (Ghazvininejad et al. 2018; Zhou et al. 2018). Our work follows the similar motivation and uses the output of IR systems as the additional knowledge source.

Combination of IR and Generative models To combine IR and generative models, early work (Qiu et al. 2017) tried to re-rank the output from both models. However, the performance of such models is limited by the capacity of individual methods. Most related to our work, (Song et al. 2016; Weston, Dinan, and Miller 2018) and (Wu et al. 2018) encoded the retrieved result into distributed representation and used it as the additional conditionals along with the standard query representation. While the former two only used the target side of the retrieved pairs, the latter took advantages of both sides. In a closed domain conversation setting, (Pandey et al. 2018) further proposed to weight different training instances by context similarity. Our model differs from them in that we take an extra intermediate step for skeleton generation to filter the retrieval information before use, which shows the effectiveness in avoiding erroneous copy in our experiments.

Multi-step Language Generation Our work is also inspired by recent success of decomposing an end-to-end language generation task into several sequential sub-tasks. For document summarization, Chen and Bansal (2018) first select salient sentences and then rewrite them in parallel. For sentiment-to-sentiment translation, Xu et al. (2018) first use a neutralization module to remove emotional words and then add sentiment to the neutralized content. Not only does their decomposition improve the overall performance, but also makes the whole generation process more interpretable. Our skeleton-to-response framework also sheds some light on the use of retrieval memories.

Experiments

Data

We use the preprocessed data in (Wu et al. 2018) as our test bed. The total dataset consists of about 20 million single-turn query-response pairs collected from Douban Group[3]. Since similar contexts may correspond to totally different responses, the training quadruples (q, r, q', r') for IR-augmented models are constructed based on response similarity. All response are indexed by Lucene.[4] For each (q, r) pair, top 30 similar responses with their corresponding contexts are retrieved $\{(q_i', r_i')\}_{i=1}^{30}$. However, only those satisfying $0.3 \leq Jaccard(r, r_i') \leq 0.7$ are leveraged for training, where $Jaccard$ measures the Jaccard distance. The reason for the data filter is that nearly identical responses drive the model to do simple copy while distantly different responses make the model ignore the retrieval input. About 42 million quadruples are obtained afterward.

For computational efficiency, we randomly sample 5 million quadruples as training data for all experiments. The test set consists of 1,000 randomly selected queries that are not in our training data.[5] For a fair comparison, when training a generative model without the help of IR, the quadruples are split to pairs.

Model Details

We implement the skeleton generator based on a bidirectional recurrent neural network with 500 LSTM units. We concatenate the hidden states from both directions. The word embedding size is set to 300. For the response generator, the encoder for queries, the encoder for skeletons and the decoder are three two-layer recurrent neural network with 500 LSTM units where bother encoders are bidirectional. We use dropout (Srivastava et al. 2014) to alleviate overfitting. The dropout rate is set to 0.3 across different layers. The same architecture for the encoders and the decoder is shared across the following baseline models, if applicable.

Compared Methods

- **Seq2Seq** the standard attention-based RNN encoder-decoder model (Bahdanau, Cho, and Bengio 2014).

- **MMI** SEQ2SEQ with Maximum Mutual Information (MMI) objective in decoding (Li et al. 2016a). In practice, an inverse (response-to-query) SEQ2SEQ model is used to rerank the N-best hypothesizes from the standard SEQ2SEQ model (N equals 100 in our experiments).

- **EditVec** the model proposed in (Wu et al. 2018), where the edit vector z is used directly at each decoding step by concatenating it to the word embeddings.

- **IR** the Lucene system is also used a benchmark.[6]

- **IR+rerank** rerank the results of **IR** by **MMI**.

Besides, We use **JNT** to denote our model with joint integration, and **CAS** for our model with cascaded integration. To validate the usefulness of the proposed skeletons. We design a response generator that takes an intact retrieval response as

---

[3]https://www.douban.com/group

[4]https://lucene.apache.org/core/

[5]Note the retrieval results for test data are based on query similarity, and no data filter is adopted.

[6]Note IR selects response candidates from the entire data collection, not restricted to the filtered one.

model	human score	dist-1	dist-2
IR	2.093	**0.238**	**0.723**
IR+rerank	2.520	0.208	0.586
Seq2Seq	2.433	0.156	0.336
MMI	2.554	0.170	0.464
EditVec	2.588†	0.154	0.394
SKP	2.581	0.152	0.406
JNT	2.612†	0.147	0.377
CAS	**2.747**	0.156	0.411

Table 1: Response performance of different models. Sign tests on human score show that the CAS is significantly better than all other methods with p-value < 0.05, and the p-value < 0.01 except for those marked by †.
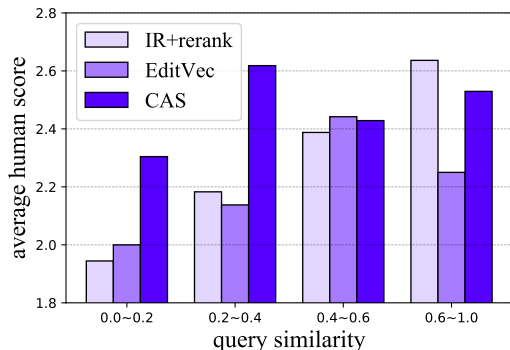

Figure 3: Response quality v.s. query similarity.[7]

its skeleton input (i.e., to completely skip the skeleton generation step), denoted by **SKP**.

There are some other IR-augmented models using standard SEQ2SEQ models as SKP. Weston, Dinan, and Miller (2018)used a rule to select either the generated response or the retrieved response as output, while we would like to focus on improving the quality of generated responses. Pandey et al. (2018) concentrated on closed domain conversations, their hierarchical encoder is not suitable for our open domain setting. We thus omit the empirical comparison with them.

Evaluation Metrics

Our method is designed to promote the informativeness of the generative model and alleviate the inappropriateness problem of the retrieval model. To measure the performance effectively, we use human evaluation along with two automatic evaluation metrics.

- **Human evaluation** We asked three experienced annotators to score the group of responses (the best output of each model) for 300 test queries. The responses are rated on a five-point scale. A response should be scored 1 if it can hardly be considered a valid response, 3 if it is a valid but not informative response, 5 if it is a informative response, which can deepen the discussion of the current topic or lead to a new topic. 2 and 4 are for decision dilem-

model	P	R	F_1	Acc.
JNT	0.32	0.61	0.42	0.60
CAS	0.50	0.86	0.63	0.76

Table 2: Performance of skeleton generator.

mas.

- **dist-1 & dist-2** It is defined as the number of unique unigrams (**dist-1**) or bi-grams (**dist-2**) dividing by the total number of tokens, measuring the diversity of the generated responses (Li et al. 2016a). Note the two metrics do not necessarily reflect the response quality as the target queries are not taken into consideration.

Response Generation Results

The results are depicted in Table 1. Overall, both of our models surpass all other methods, and our cascaded model (CAS) gives the best performance according to human evaluation. The contrast with the SKP model illustrates that the use of skeletons brings a significant performance gain.

According to the dist-1&2 metrics, the generative models achieve significantly better diversity by the use of retrieval results. The retrieval method yields the highest diversity, which is consistent with our intuition that the retrieval responses typically contain large amount of information though they are not necessarily appropriate. The model of MMI also gives strong diversity, yet we find that it tends to simply repeat the words in queries. By removing the words in queries, the dist-2 of MMI and CAS become 0.710 and 0.751 respectively. This indicates our models are better at generating new words.

To further reveal the source of performance gain, we study the relation between response quality and query similarity (measured by the Jaccard similarity between the current query and the retrieved query). Our best model (CAS) is compared with the strong IR system (IR-rerank) and the previous state-of-the-art (EditVec) in Fig. 3. The CAS model significantly boosts the performance when query similarity is relatively low, which indicates that introducing skeletons can alleviate erroneous copy and keep a strong generalization ability of the underlying generative model.

More Analysis of Our Framework

Here, we present further discussions and empirical analysis of our framework.

Generated Skeletons Although generating skeletons is not our primary goal, it is interesting to assess the skeleton generation. The word-level precision (P), recall (R), F_1 score (F_1) and accuracy (Acc.) of the well-trained skeleton generators are reported in Table 2, taking the proxy skeletons as golden references.

Table 3 shows some skeleton-to-response examples of the CAS model and a case study among different models. In the

_____

[7]We merge the ranges $[0.6, 0.8]$ and $[0.8, 1.0]$ due to the sparsity of highly similar pairs.

Query	善良的妹子会幸福一生的 May you kind girls have a happy life	好的到时候有时间骑车出来玩嘛， 我觉得在大理最适合骑车玩了 OK, Let's ride when free. It is Dali.	把你联系方式给我 Tell me your contact
Retrieved query	谢谢善良的妹子 Thanks to the kind girl	改天一起出来骑车吧,我们都喜欢玩单车 Let's ride when free, as all of us like riding	把你姐姐联系方式给我 Tell me your sister's contact
Retrieved response	我是汉子哈哈 I am a boy, LOL	ok 加微信abcdefg ok add my wechat abcdefg	我姐已婚 My sister is married.
Skeleton	_汉子_ _boy_	ok加微信_ OK, add my wechat _	_姐_已婚 _ sister _ married.
Generated response (CAS)	哈哈哈，我是善良的**汉子** LOL, I am a kind **boy**	你**加**我**微信**吧，我也在大理 **Add** my **wechat**, I am in Dali, too	我**已婚**哈哈 I am **married**, LOL
IR+rerank	谢谢嘛 Thanks	ok 加微信: abcdefg OK, add my Wechat: abcdefg	我姐已婚 My sister is married.
MMI	我不是善良的妹纸 I am not a kind girl	我也想骑在大理 I want to ride Dali, too	你怎么知道我联系方式 How did you know my contact
EditVec	谢谢，善良的妹子也会幸福的 Thanks, a kind girl will be happy too	有时间可以去玩 Let's go when free	哈哈哈哈好啊 LOL, Ok!
JNT	我不是善良的妹子 I am not a kind girl	我也在大理 I am in Dali, too	我不是联系的 I am not the contact person

Table 3: Upper: Skeleton-to-response examples of the CAS model. Lower: Responses from different models are for comparison.

leftmost example in Table 3, the MMI and the EditVec simply repeat the query while the retrieved response is weakly related to the query. Our CAS model extracts a useful word 'boy' from the retrieved response and generate a more interesting response. In the middle example, the MMI response make less sense, and some private information is included in the retrieved response. Our CAS model removes the privacy without the loss of informativeness, while the outputs by other models are less informative. The rightmost case shows that our response generator is able to recover the possible mistakes made by the skeleton generator.

Retrieved Response v.s. Generated Response To measure the extent that the generative models are paying attention to and copying the retrieval, we compute the edit distances between generated responses and retrieved responses. As shown in Fig. 4, in the comparison between the SKP and other models, the use of skeletons makes the generated response deviate more from its prototype response. Ideally, when the retrieved context is very similar to the current query, the changes between the generated response and the prototype response should be minor. Conversely, the changes should be drastic. Fig. 4 also shows that our models can learn this intuition.

Single v.s. Multiple Retrieval Pair(s) For a given query q, the retrieval pair set R_q could contain multiple query-response pairs. We investigate two ways of using it under the CAS setting.

- **Single** For each query-response pair $(q'_i, r'_i) \in R_q$, a response $\hat{r}_i$ is generated solely based on q, and (q'_i, r'_i). The resulted responses are reranked by generation probability.

- **Multiple** The whole retrieval set R_q is used in a single run. Multiple skeletons are generated and concatenated in the response generation stage.
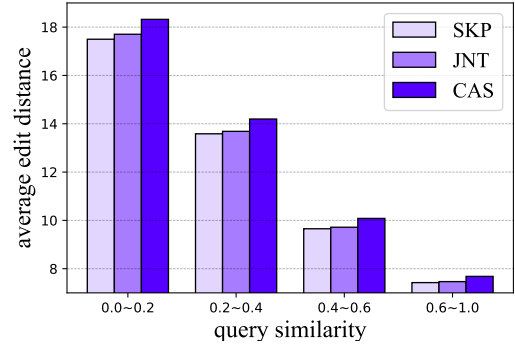

Figure 4: Changes between retrieved and generated responses v.s. query similarity.

setting	human score	dist-1	dist-2
Single	2.747	0.156	0.411
Multiple	1.976	0.178	0.414

Table 4: Comparison of the usages of the retrieval set.

The results are shown in Table 4. We attribute the failure of **Multiple** to the huge variety of the retrieved responses. The response generator receives many heterogeneous skeletons, yet it has no idea which to use. It remains an open question on how to effectively use multiple retrieval pairs for generating one single response and we leave it for future work.

Conclusion

In this paper, we proposed a new methodology to enhance generative models with information retrieval technologies for dialogue response generation. Given a dialogue context, our methods generate a skeleton based on historical

responses that respond to a similar context. The skeleton serves as an additional knowledge source that helps specify the response direction and complement the response content. Experiments on real world data validated the effectiveness of our method for more informative and appropriate responses.

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Cao, K., and Clark, S. 2017. Latent variable dialogue models and their diversity. In *EACL*, 182–187.

Chen, Y.-C., and Bansal, M. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *ACL*.

Ghazvininejad, M.; Brockett, C.; Chang, M.-W.; Dolan, B.; Gao, J.; Yih, W.-t.; and Galley, M. 2018. A knowledge-grounded neural conversation model. In *AAAI*, 5110–5117.

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*.

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

Li, J.; Galley, M.; Brockett, C.; Spithourakis, G. P.; Gao, J.; and Dolan, B. 2016b. A persona-based neural conversation model. In *ACL*, 994–1003.

Li, D.; He, X.; Huang, Q.; Sun, M.-T.; and Zhang, L. 2018. Generating diverse and accurate visual captions by comparative adversarial learning. *arXiv preprint arXiv:1804.00861*.

Luong, T.; Pham, H.; and Manning, C. D. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*, 1412–1421.

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

Qiu, M.; Li, F.-L.; Wang, S.; Gao, X.; Chen, Y.; Zhao, W.; Chen, H.; Huang, J.; and Chu, W. 2017. Alime chat: A sequence to sequence and rerank based chatbot engine. In *ACL*, 498–503.

Serban, I. V.; Sordoni, A.; Bengio, Y.; Courville, A. C.; and Pineau, J. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, volume 16, 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*, 1577–1586.

Shen, X.; Su, H.; Li, Y.; Li, W.; Niu, S.; Zhao, Y.; Aizawa, A.; and Long, G. 2017. A conditional variational framework for dialog generation. In *ACL*, 504–509.

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.-Y.; Gao, J.; and Dolan, B. 2015. A neural network approach to context-sensitive generation of conversational responses. In *NAACL*, 196–205.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.

Vinyals, O., and Le, Q. 2015. A neural conversational model. In *ICML (Deep Learning Workshop)*.

Wagner, R. A., and Fischer, M. J. 1974. The string-to-string correction problem. *Journal of the ACM (JACM)* 21(1):168–173.

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

Weston, J.; Dinan, E.; and Miller, A. H. 2018. Retrieve and refine: Improved sequence generation models for dialogue. *arXiv preprint arXiv:1808.04776*.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

Wu, Y.; Wei, F.; Huang, S.; Li, Z.; and Zhou, M. 2018. Response generation by context-aware prototype editing. *arXiv preprint arXiv:1806.07042*.

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

Xu, J.; Sun, X.; Zeng, Q.; Ren, X.; Zhang, X.; Wang, H.; and Li, W. 2018. Unpaired sentiment-to-sentiment translation: A cycled reinforcement learning approach. In *ACL*, 675686.

Zhao, T.; Lee, K.; and Eskenazi, M. 2018. Unsupervised discrete sentence representation learning for interpretable neural dialog generation. In *ACL*, 1098–1107.

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

Zhou, H.; Young, T.; Huang, M.; Zhao, H.; Xu, J.; and Zhu, X. 2018. Commonsense knowledge aware conversation generation with graph attention. In *IJCAI*, 4623–4629.