Retrieval-guided Dialogue Response Generation via a Matching-to-Generation Framework

Deng Cai[1*] **Yan Wang**[2] **Wei Bi**[2] **Zhaopeng Tu**[2] **Xiaojiang Liu**[2] **Shuming Shi**[2]
[1]The Chinese University of Hong Kong, [2]Tencent AI Lab
thisisjcykcd@gmail.com
{brandenwang,victoriabi,zptu,kieranliu,shumingshi}@tencent.com

Abstract

End-to-end sequence generation is a popular technique for developing open domain dialogue systems, though they suffer from the *safe response problem*. Researchers have attempted to tackle this problem by incorporating generative models with the returns of retrieval systems. Recently, a skeleton-then-response framework has been shown promising results for this task. Nevertheless, how to precisely extract a skeleton and how to effectively train a retrieval-guided response generator are still challenging. This paper presents a novel framework in which the skeleton extraction is made by an interpretable matching model and the following skeleton-guided response generation is accomplished by a separately trained generator. Extensive experiments demonstrate the effectiveness of our model designs.

1 Introduction

Sequence-to-sequence (seq2seq) neural models (Shang et al., 2015; Vinyals and Le, 2015; Sordoni et al., 2015; Serban et al., 2016; Li et al., 2016a) have been popular for single-turn dialogue response generation. However, many of the generated responses (e.g., "I don't know" and "I think so") appear to be generic and dull (safe response problem) (Li et al., 2016a). This problem is avoided in traditional retrieval systems (Ji et al., 2014; Hu et al., 2014) by preceding the selection of informative and engaging responses.

It is of interest to benefit from both the generalization capacity of the seq2seq models and the information richness of the retrieved responses. Following the standard encoder-decoder framework, early attempts have either used an extra encoder for the retrieved response (Song et al., 2016; Pandey et al., 2018; Wu et al., 2019) or a unified encoder for the concatenation of the query and the retrieved response (Weston et al., 2018). To prevent the inflow of erroneous information, Cai et al. (2019) proposed a general framework that first extracts a skeleton from the retrieved response and then generates the response based on the extracted skeleton. Despite their differences, a common issue is that the generation model easily learns to ignore the retrieved response entirely and collapses to a vanilla seq2seq model. As shown in Figure 1, this happens with improper training instances. Given the large space of possible responses, it happens frequently that a retrieved response (extracted skeleton) is suitable for responding to the query, but inconsistent with the current target response.[1] The generation model is thus mistakenly led to be inclined to neglect the retrieval.

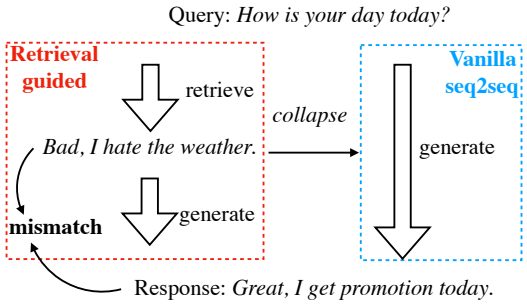

Figure 1: The common problem for training a retrieval-guided generation model in previous work. The model is forced to neglect the retrieved response even though it is a proper response, due to the mismatch between the retrieved response and the target response.

To address the above problem, we present the matching-to-generation method, a more flexible

[*] This work was mainly done when Deng Cai was an intern at Tencent AI Lab.

[1] Previous studies (Weston et al., 2018; Wu et al., 2019; Cai et al., 2019) alleviated the problems by putting hard constraints on the data, which, however, greatly reduces the amount of usable data (e.g., Cai et al. (2019) required the Jaccard distance between the retrieved response and the target response should be in the range [0.3, 0.7]).

framework for retrieval-guided response generation. This framework consists of an interpretable matching model for skeleton extraction and a skeleton-guided response generator for response generation. One novel characteristic of our proposed framework is that the training of the skeleton extractor (i.e., the matching model) and the response generator is decoupled, yet they work cooperatively under the help of a retrieval system. Figure 2 depicts the training and inference procedures of our framework. During training, the skeleton-guided response generator is trained in a similar manner as the denoising autoencoder (Vincent et al., 2008), where the model learns to recover an input pattern that is partially corrupted. Specifically, we employ a random mechanism for generating the skeletons used for training. The generated skeletons are extracted from their corresponding responses with some deliberate disturbance. In this way, we circumvent the aforementioned problem of improper training instances in previous work. Meanwhile, the random mechanism also simulates the actual inference environment where the quality of the input skeleton varies among different queries due to the instability of the retrieval system and the skeleton extractor. The diversity of the training skeletons helps produce a robust response generator that is capable of handling different situations.

Note the separation of the training of skeleton extraction and response generation requires an additional training objective for the skeleton extractor. Given there is no explicit response skeleton in general query-response pairs for training, we propose to use an interpretable matching model for *matching skeleton* extraction. We consider that the matching skeleton for a given query-response pair should be the sub-sequence of the response that is particularly useful in matching the query. The designed interpretable matching model is able to reveal the fine-grained matching scores at token-level whereas it is trained by ordinary query-response pairs.

Experiments show that our method significantly improves the informativeness of the generated responses as well as their relevance to the corresponding queries. In addition, we conduct extensive ablation studies to quantify the improvement from different model designs.

To summarize, our contributions are as follows:

- We propose a flexible framework for retrieval-guided dialogue response generation. The training of our approach is independent of the underlying retrieval system.
- We propose an interpretable matching model for matching skeleton extraction.
- We propose to train a skeleton-guided response generator that can handle skeletons with different qualities.

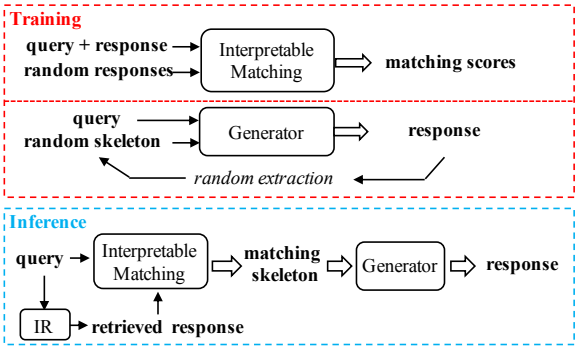

Figure 2: Flow charts during training and inference.

2 Models

The whole framework consists of two components: an interpretable matching model and a skeleton-guided response generator. During inference, the matching model is used to derive a matching skeleton by explicitly selecting a subsequence of a retrieved response. The response generator then takes the generated skeleton as an additional input and makes necessary editions to obtain a complete and appropriate response.

2.1 Interpretable Matching Model

The goal of the interpretable matching model is to reveal token-level matching information between a query-response pair thus a matching skeleton can be derived from the response. However, the training of the matching model does not rely on such fine-grained annotations. Instead, it is trained to estimate the sequence-level quality of a response for a given query, as an ordinary query-response matching model. The key is that the sequence-level matching score can be decomposed into a set of token-level scores, which will be illustrated later.

The overall architecture of our matching model is illustrated in Figure 3. It consists of two encoders, one for the query and one for the re-

sponse. Both encoders are based on the Transformer architecture (Vaswani et al., 2017). For a query $q = (q_1, q_2, \ldots, q_n)$ and a response $r = (r_1, r_2, \ldots, r_m)$, where n and m are the query length and the response length respectively, we first insert a special token at the beginning of each input sequence. The transformer encoders results in two sequences of hidden state vectors $\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_n$ and $\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_m$, where $\mathbf{q}_0$ and $\mathbf{r}_0$ are considered as the aggregate summary for the query and the response respectively.

We then use self-attention mechanism for acquiring the final query representation $\mathbf{x}_q$ and the final response representation $\mathbf{x}_r$. For instance, to compute the response representation $\mathbf{x}_r$, we use the sequence-level summary $\mathbf{r}_0$ for weighting the different parts of the input response. First, the sequence-level summary $\mathbf{r}_0$ is projected to another vector space by a linear transformation:

$$\mathbf{r}^w = W^w \mathbf{r}_0 + b^w$$

where $\mathbf{r}^w$ is the weight vector, and W^w and b^w are learnable parameters. The attention score ω_i of the i-th token in the response is then computed as a dot-product between the weight vector $\mathbf{r}^w$ and the token representation $\mathbf{r}_i$:

$$\omega_i = \frac{\exp(\mathbf{r}^w \cdot \mathbf{r}_i)}{\sum_{k=1}^{m} \exp(\mathbf{r}^w \cdot \mathbf{r}_k)}$$

The response representation $\mathbf{x}_r$ is calculated as the weighted sum of the Transformer encoder outputs as well as their initial vector representations (i.e., the sum of tokens and position embeddings)[2]:

$$\mathbf{x}_r = \sum_{k=1}^{m} \omega_i (\mathbf{r}_i + \mathbf{e}_{r_i})$$

The self-attention mechanism for the query has the identical architecture but uses a different set of parameters. Finally, the pair-wise score is calculated by a bilinear function of $\mathbf{x}_q$ and $\mathbf{x}_r$:

$$s(q, r) = {\mathbf{x}_q}^T W^s \mathbf{x}_r$$

where W^s is a trainable parameter. The above equation can be rewritten by a decomposition of

[2]We found that adding the initial vector representations $\mathbf{e}_{r_i}$ to be critical in keeping the weighted elements reflect the corresponding local information. Without this operation, the Transformer encoder outputs $\mathbf{r}_i$ tends to be constant regardless of the position i, which indicates that the point information about a specific input part is overwhelmed by the global information.

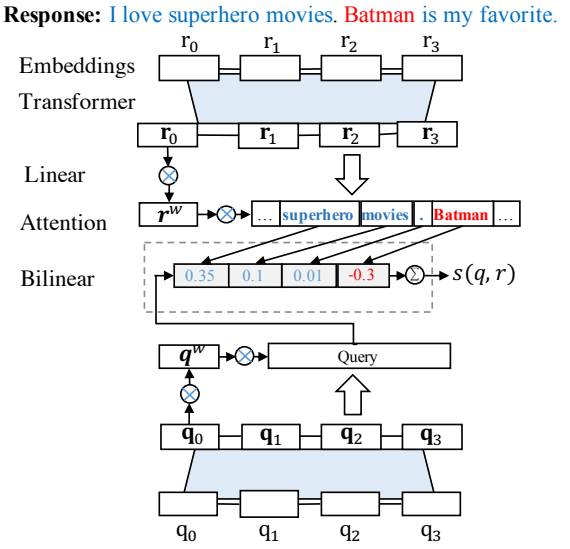

Figure 3: Architecture of the interpretable matching model. The query and response are encoded separately. We compute all attention values for each token simultaneously. The final score is computed by a bilinear function between query representation and response representation. Different colors indicate the different values of the local scores.

the ingredients of $\mathbf{x}_r$:

$$\begin{aligned} s(q, r) &= {\mathbf{x}_q}^T W^s \mathbf{x}_r \\ &= {\mathbf{x}_q}^T W^s \sum_{k=1}^{m} \omega_k (\mathbf{r}_k + \mathbf{e}_{r_k}) \\ &= \sum_{k=1}^{m} \omega_k {\mathbf{x}_q}^T W^s (\mathbf{r}_k + \mathbf{e}_{r_k}) \end{aligned}$$

Let $s_k = {\mathbf{x}_q}^T W^s (\mathbf{r}_k + \mathbf{e}_{r_k})$, we arrive at:

$$s(q, r) = \sum_{k=1}^{m} \omega_k s_k$$

Note that for a given query, ω_k and s_k are functions of the response r and the position index k only. According to the formulation, we see that s_k and ω_k are largely impacted by the local information at r_k. Therefore, s_k, ω_k can be interpreted as the *local matching score* and the *local importance*, respectively, followed by that $s(q, r)$ is a weighted sum of all local scores.

Once the matching model has been well-trained, we can use ω_k and s_k to identify the most informative and relevant parts of a retrieved response. In experiments, we show a simple heuristic rule can effectively pick up the skeletons.

2.2 Skeleton-guided Response Generator

The skeleton-guided response generator is devised for generating a fluent and adequate response based on the current query and an input skeleton. To ensure the skeleton-guided response generator does make use of the input skeleton, we extract the training skeleton from the ground-truth response by some randomized strategies. To prevent the response generator from mindlessly copying, we deliberately vary the length and the quality of the training skeletons to create a diverse set of training instances. We note the response generator behaves like a denoising autoencoder (Vincent et al., 2008) with an extra input, i.e. the query. In this way, we learn a robust response generator that is compatible with different types of skeletons.

Specifically, for any golden query-response pair (q, r), we randomly generate a training skeleton through the following procedures.[3]

- All stop words in r are masked in advance. The rest tokens are masked at a mask rate γ. 90% of the time, γ is set to 0.7. 10% of the time, γ is uniformly sampled in the range of $[0, 1]$.
- Instead of always replacing the masked token with a special placeholder token, 20% of time, we replace the token with a random word uniformly sampled from the total vocabulary.
- At a chance of 10%, we randomly shuffle the word order in the training skeleton.

The response generator consists of one encoder for the query q, one encoder for the skeleton s and one decoder for the response r, all implemented by LSTM networks (Hochreiter and Schmidhuber, 1997). The decoder interacts with the two encoders through two separate attention mechanisms accordingly.

2.3 Training

The matching model and the response generator are trained separately. Previous studies (Shang et al., 2018; Tao et al., 2018; Mou et al., 2016) formulated the training of matching models as binary classification learning, where negative sampling is used to free human annotation. Specifically, for query q and golden response r^+, a negative response r^- can be randomly sampled from other responses in the training set. We extend the binary classification setting into a learning-to-rank fashion for improved performance. Concretely, at each training mini-batch, we randomly sample M query-response pairs. Then we compute the matching scores between all combinations of queries and responses in the mini-batch. As a result, all these scores form a scoring matrix $\mathbf{S} \in \mathbb{R}^{M \times M}$, where $\mathbf{S}_{ij}$ is the score between the i-th query and the j-th response.

Inspired by Henderson et al. (2017); Lin et al. (2017), we use softmax to compute the ranking scores for candidate responses. Intuitively, for each query, the matching model should give the highest score to the golden response over other $M - 1$ responses (i.e., always rank the golden response at the first place). Thus, we define the training loss as

$$L(\theta) = -\sum_{k=1}^{M} \log \text{softmax}(\mathbf{S}_{k:})_k \qquad (1)$$

where $\mathbf{S}_{k:}$ is k-th row of $\mathbf{S}$. Label smoothing (Szegedy et al., 2016) of value $\epsilon_{ls} = 0.1$ is used to improve the performance. Note although there are $M \times M$ scores to compute, each query and each response only needs to be modeled once thanks to the independent encoding of $\mathbf{x}_q$ and $\mathbf{x}_r$. Experiments show that the ranking scheme outperforms the binary classification scheme by a large margin when evaluated by hits@1 metric with 127 randomly sampled responses.

The response generator is trained by the standard maximum likelihood estimate.

2.4 Discussion

We note that the most related work is Cai et al. (2019) that also employs a pipeline approach for skeleton extraction and response generation. However, there are some major distinctions in our framework. First, their skeleton extractor is pre-trained by the lexical overlap between the retrieved response and the golden response. However, it is not a proper objective since the mismatch between the retrieved response and the golden response does not imply a mismatch to the target query. In contrast, our interpretable matching model allows extracting a more precise skeleton in semantics. Second, the training of the response generator relies on the output of the learned skeleton extractor, which is by no means aimed for generating the current response, causing a trained

[3] We did not tune much the hyper-parameters in the random strategies as the given setting just works fine.

generator to severely ignore the skeleton. Differently, our response generator is trained with target-specific skeletons.

3 Experiments

3.1 Dataset and Evaluation Metrics

We use a single-turn conversation dataset collected from popular Chinese social websites such as Douban and Weibo.[4] The dataset contains about six millions query-response pairs. Throughout all experiments, the retrieval system we adopted is a publicly available chatbot API.[5] The related resources can be found at https://github.com/jcyk/seqgen.

It has been argued that existing automatic metrics such as BLEU and METEOR cannot authentically reflect the quality of dialog response. Thus, the main evaluation is done by human annotators. Specifically, we evaluate the quality of a response on three criteria: informativeness, relevance, and fluency. Each aspect is rated on a five-point scale, where 1, 3 and 5 indicate unacceptable, moderate and excellent performance respectively. 2 and 4 are used by annotators in unsure cases. A set of 300 different query samples are used for evaluation. We recruit five experienced annotators and take the average score among them. Besides, we also use *dist-1*/*dist-2* (Li et al., 2016a) to examine a model's ability for generating diverse responses. which is the number of distinct unigrams/bi-grams divided by the total number.

3.2 Compared Methods

To show the effectiveness of our proposed methods, we compare it with the following methods.

- *Retrieval* The underlying retrieval system used in our experiments.
- *Seq2Seq* The basic Seq2Seq model (Bahdanau et al., 2014; Luong et al., 2015) that only takes the query as input.
- *Seq2Seq-MMI* A variant of the basic Seq2Seq model that uses Maximum Mutual Information (MMI) for filtering out generic responses (Li et al., 2016a). Concretely, a response-to-query Seq2Seq model is trained and used to rerank the outputs of the top-100 responses of *Seq2Seq*.
- *RetrieveNRefine*$^{++}$ The best performing model used in Weston et al. (2018), which appends the retrieved response to the query in a basic Seq2Seq model.[6] The model's output will be overwritten by the retrieved response once they have a large word overlap (Jaccard distance > 0.6).
- *EditVec* The model proposed in Wu et al. (2019). In addition to the retrieved response, the lexical difference (insert words and delete words) between the query and the retrieved query is also encoded (in a so-called edit vector) to feed the decoder.
- *Skeleton-Lex* The best method presented in Cai et al. (2019). We refer to it as *Skeleton-Lex* because its skeleton extractor is pre-trained by the lexical overlap between the retrieved response and the golden response.

3.3 Implementation Details

For encoders and decoders in all above baselines, they are implemented by LSTM networks (bidirectional for encoders and unidirectional for decoders) (Hochreiter and Schmidhuber, 1997) with the number of layers and hidden size equal to 2 and 500. The word embeddings are randomly initialized, of which the dimension is 300. Our response generator follows the same settings. The skeleton extractor is implemented by 2-layer Transformer encoder (Vaswani et al., 2017), of which the number of heads and hidden size is 8 and 512. In experiments, we use a simple heuristic rule for extracting skeletons. First, we remove all words with a negative local score s_k. Then we compute the average score of the rest part. Lastly, words with a score below the average are also removed.

As the retrieval system can potentially return a large set of results, we allow retrieval-guided generation models (both baselines and ours) make use of the top-10 retrieved results both in training and testing. Therefore, during testing, 10 responses are generated for each query. They are then ranked by the matching model proposed in Section 2.1 and the highest-scored one is used for evaluation.[7]

[4] Douban https://www.douban.com/ and Weibo https://www.weibo.com/

[5] https://ai.qq.com/product/nlpchat.shtml

[6] We take a slightly different approach by treating the two inputs as independent sources to do attention over, as it is also suggested by the original authors.

[7] We train another matching model (not the one we used as skeleton extractor).

Models	Informativeness	Relevance	Fluency	Dist-1(%)	Dist-2(%)
Retrieval	2.65 (0.90)†	2.58 (0.86)	2.96 (0.72)	**49.10**	**84.19**
Seq2Seq	2.01 (0.65)	2.58 (0.53)	2.71 (0.43)	30.38	54.52
Seq2Seq-MMI	2.47 (0.70)	2.79 (0.67)	2.99 (0.61)	30.98	62.85
RetrieveNRefine^{++}	2.30 (0.79)	2.62 (0.63)	2.82 (0.51)	29.83	61.07
EditVec	2.29 (0.61)	2.62 (0.60)	2.83 (0.47)	35.30	67.57
Skeleton-Lex	2.45 (0.61)	2.80 (0.56)	2.99 (0.46)	25.70	56.61
Ours	**2.69** (0.87)	**3.11** (0.55)	**3.20** (0.55)	**49.01**	**80.36**

Table 1: Human scores on response quality, depicted in three aspects: informativeness, relevance, and fluency, with standard deviation in parentheses. Sign tests on human scores show that our method is significantly better than all other methods with p-value <0.01 with the only exception marked by †. We also present dist-1 and dist-2 for diversity assessment.

3.4 Main Results

The evaluation results are given in Table 1. They show that our method outperforms all baseline methods in all three human evaluation aspects. Surprisingly, the informativeness score is even slightly better than the underlying retrieval system, which indicates the retrieved information has been effectively utilized. It can also be verified by the automatic metrics (dist-1 and dist-2), our generation model is the only one that achieves close performance to that of the retrieval system.

For the relevance score, the retrieval-independent *Seq2Seq-MMI* establishes a strong baseline. As for retrieval-guided generation, skeleton-guided methods are better than those who use completely retrieved responses, which confirms that the introduction of the intermediate skeleton prevents the inflow of irrelevant information. Furthermore, our method advances the performance of *Skeleton-Lex* by a large margin, which partly demonstrates that the skeletons extracted by our deep semantic matching model are more precise.

For fluency, our method also achieves much better performance than all baseline methods. We attribute the remarkable improvement to the unique training fashion for our response generator. During training, our response generator receives a diverse set of probably noisy skeletons, which impels it to learn the error correction and better language organization.

3.5 More Analysis

To further quantify the contributions made by different components in our model, we turn to ablation tests. Generally, we try to substitute each component of our model with other possible counterparts. The detailed analysis is given below.

Skeletons	Info.	Relevance	Fluency
Ours	2.69	3.11	3.20
Lexical	2.62	2.92	3.05
keywords	2.56	2.90	3.03
PMI	2.53	2.88	3.02

Table 2: Ablation study on the skeleton extractor. Info. is short for informativeness in this and following tables.

First, we would like to see if the matching skeleton extracted by our interpretable architecture is beneficial. In order to examine this, we replace our skeleton extractor by several different approaches.

- *Lexical* We use the skeletons extracted by the skeleton extractor in *Skeleton-Lex*.
- *PMI* Point mutual information (PMI) is a popular measure used for finding collocations and associations between words. We compute the PMI between query word and response word through statistics on the training corpus. For a word in the retrieved response, we score it by the sum of the PMIs between it and all words in the target query. Words with the highest scores form the skeleton.
- *Keywords* We generate a skeleton by preserving the most informative words in the retrieved response. Specifically, the words with the highest TF-IDF values are preserved and the others are removed.

For a fair comparison, the lengths of the skeletons (the number of preserving words) generated by *PMI* and *Keywords* are kept as the same with the one generated by our skeleton extractor. In this sense, the comparison with the last two approaches shows how good the token-level score s_k

Model Variants	Info.	Rel.	Flu.
Ours	2.69	3.11	3.20
Matching + C19's RG	2.46	2.72	2.89
C19's SE + Generator	2.62	2.92	3.05
Cai et al. (2019)	2.45	2.80	2.99

Table 3: A systematic comparison of different component combinations, where C19's SE and C19's RG are short for Cai et al. (2019)'s skeleton extractor and response generator respectively.

is in selecting the most useful words, compared to statistical values such as TF-IDF and PMI.

The result is in shown Table 2. As seen, both two learnable skeleton extractors give better results than non-parametric methods, indicating the task of skeleton extraction is non-trivial and requires deep reasoning. Our semantic-inspired model is far ahead of others in all aspects, while *Lexical* only has a notable improvement in informativeness compared to statistical methods. This suggests that the skeleton extracted by *Lexical* has a relatively low precision, leading to moderate relevance. In addition, it might be a little bit surprising to see that *PMI* and *keywords* give almost the same performance on all three metrics, telling a given query is not that necessary. However, we found lots of skeletons proposed by *PMI* are identical to those of *keywords*. We attribute it to that the keywords in r are often also the keywords in differentiating its context.

To test the ability of the skeleton-based response generator, we use the existing alternative trained in *Skeleton-Lex*, which also takes a skeleton and input query as input. The result displays on how good our response generator is at transforming a skeleton to a proper response.

The results are shown in the first block of Table 3. We see a clear decline in performance after switching to the response generator of *Skeleton-Lex*. We conjecture that the big gap is caused by that their response generator is trained with the output of their skeleton extractor, thus it is highly biased to their specific skeleton extractor and cannot work well with others. This result motivates us to present a systematic examination of different component combinations, as shown by the full content of Table 3. As seen, our response generator is less sensitive to the underlying skeletons. The result of combining our response generator and their skeleton extractor is slightly below our full mode, but even higher than their full model.

Variants	Info.	Relevance	Fluency
ours	2.69	3.11	3.20
Seq2Seq-MMI	2.64	3.02	3.14

Table 4: Ablation study on the ranker.

Lastly, we investigate the improvement brought by our ranker. To this end, we replace our ranker with the *seq2seq-MMI* model (use the sum of the forward and backward generation probability of re-ranking). The results are shown in Table 4. As we can see, our matching model shows superior capability in selecting the best response, especially in terms of relevance.

3.6 Case Study

We also present some examples generated by different methods in Table 5. In the first case, the retrieved utterance is very specific with elaborated details. However, it is not a reasonable response due to the sudden topic drift. While other methods directly ignore the retrieved response, our method makes an informative and fluent response by using some useful details. In the second case, our skeleton extractor again successfully locates the most informative and relevant parts of the retrieved response, the response generator then chains them together and generate a meaningful response. In the third case, the retrieved response is also good but not fluent. The skeleton extracted by our skeleton extractor contains a useless word, yet our response generator can ignore the mistake and generate a fluent and comforting response. We can also see another retrieval-guided method (*EditVec*) also attempt to use the retrieved response but capture the wrong parts.

4 Related Work

Open domain dialog system has been a long goal for the NLP community since ELIZA (Weizenbaum, 1966). Early data-driven work uses information retrieval techniques (Ji et al., 2014; Hu et al., 2014). Recently, end-to-end neural sequence generation (Vinyals and Le, 2015; Serban et al., 2016; Li et al., 2016a; Sordoni et al., 2015) has attracted the most attention. A major issue of such end-to-end sequence generation method is the *safe response problem*. The generated responses tend to be universal and unengaging (e.g., "I don't know", "I think so" etc.). One of the rea-

Query	喜欢长头发的女生 I like girls with long hair
Retrieved response	可以 把 头发 后面 汤 大 卷 , 前面 是 直 刘海 , 这样 看 的 很 可爱 You can try perming the hair , hair with curly ends and straight bangs is very cute.
Ours	我也喜欢长**头发**的女生,**卷**了**直刘海**, 头发就不好看了 I also like girl with long **hair**, I think once the **straight bangs** are **curled**, it doesn't look good any more.
Seq2Seq-MMI	我也喜欢长头发的女生 I also like girls with long hair.
RetrieveNRefine^{++}	我喜欢长头发的女生 I like girls with long hair.
EditVec	我也喜欢长头发的女生 I also like girls with long hair.
Skeleton-Lex	我喜欢长头发的女生 I like girls with long hair.
Query	我提现怎么还没到 Why hasn't my withdrawal arrived yet?
Retrieved response	就是 提现 到 你 绑定 的 银行卡 上 了 Withdrawal is in the bound bank card .
Ours	你**提现**的时候**绑定**了**银行卡**吗 Have you **bounded** the **bank card** when **withdrawing** ?
Seq2Seq-MMI	我提现也到了 My withdrawal has arrived too.
RetrieveNRefine^{++}	你要支付宝干嘛 Why do you need Alipay?
EditVec	你是提现的吗 Do you want to withdraw?
Skeleton-Lex	你不是已经到了吗 Haven't you arrived ?
Query	我月经不太规律 I have irregular periods.
Retrieved response	去 看 医生 啊 . 最 好看 中医 , 挺准 的 , 别 不好意思 Go to see a doctor , best see a traditional Chinese doctor . Pretty accurate , don't be shy.
Ours	看**医生**吧,**最**简单的方法就是**中医**调理一下了 See a **doctor**, the **easiest** way is with **Chinese medicine** recuperation.
Seq2Seq-MMI	我也不规律 I am irregular too.
RetrieveNRefine^{++}	有啥不规律的 What is irregular?
EditVec	有啥不好意思的 Why you are shy?
Skeleton-Lex	我也不规律 I am irregular too.

Table 5: Example responses from different models, as well as a visualization of the skeleton extraction in our model. Darker color indicates bigger matching scores and the words being selected for skeleton are in red boxes.

sons is that for most queries, the set of possible responses is considerably large and the query alone cannot specify an informative response. Various approaches (Li et al., 2016b; Zhao et al., 2018; Xing et al., 2017; Ghazvininejad et al., 2018; Zhou et al., 2018) have been explored to address the problem.

Some previous studies have been about using the results of traditional retrieval systems for informative response generation. Song et al. (2016) introduced an extra encoder for the retrieved response. The encoder's output, together with that of the query encoder, is utilized to feed the decoder. Weston et al. (2018) simply concatenated the original query and the retrieved response as the input to the encoder. Instead of solely using the retrieved response, Wu et al. (2019) further introduced to encodes the lexical differences between the current query and the retrieved query. Besides, Pandey et al. (2018) proposed to weight different training instances by context similarity, yet their work is done in close domain conversation. The idea of editing some prototype materials rather than generating from scratch has also been explored in other text generation tasks. For examples, Guu et al. (2018) proposed a prototype-then-edit model for unconditional text generation. Wiseman et al. (2017, 2018) used either fixed template or learned templates for data-to-text generation. Xu et al. (2018) conditioned the next sentence generation on a skeleton that is extracted from the source input and the already generated text in storytelling. Also for storytelling, Clark et al. (2018) proposed to extract the entities in sentences and use them as additional input. Gu et al. (2018) uses retrieved translation as a reference to the generative translation model.

5 Conclusion

In this paper, we presented a novel framework, matching-to-generation, for retrieval-guided response generation. Our method uses an interpretable matching model for response skeleton extraction and a robust response generator for response completion. The two components are trained separately to allow more flexibility. Experiments show our method significantly outperforms several strong baselines.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Deng Cai, Yan Wang, Wei Bi, Zhaopeng Tu, Xiaojiang Liu, Wai Lam, and Shuming Shi. 2019. Skeleton-to-response: Dialogue generation guided by retrieval memory. In *NAACL*.

Elizabeth Clark, Yangfeng Ji, and Noah A. Smith. 2018. Neural text generation in stories using entity representations as context. In *NAACL*, pages 2250–2260.

Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih, and Michel Galley. 2018. A knowledge-grounded neural conversation model. In *AAAI*, pages 5110–5117.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor OK Li. 2018. Search engine guided neural machine translation. In *AAAI*.

Kelvin Guu, Tatsunori B Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *Transactions of the Association for Computational Linguistics*, 6:437–450.

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. *arXiv preprint arXiv:1705.00652*.

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

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *NIPS*, pages 2042–2050.

Zongcheng Ji, Zhengdong Lu, and Hang Li. 2014. An information retrieval approach to short text conversation. *arXiv preprint arXiv:1408.6988*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *NAACL*, pages 110–119.

Jiwei Li, Michel Galley, Chris Brockett, Georgios P Spithourakis, Jianfeng Gao, and Bill Dolan. 2016b. A persona-based neural conversation model. In *ACL*, pages 994–1003.

Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. 2017. Adversarial ranking for language generation. In *NIPS*, pages 3155–3165.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*, pages 1412–1421.

Lili Mou, Yiping Song, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2016. Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation. In *COLING*, pages 3349–3358.

Gaurav Pandey, Danish Contractor, Vineet Kumar, and Sachindra Joshi. 2018. Exemplar encoder-decoder for neural conversation generation. In *ACL*, pages 1329–1338.

Iulian Vlad 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*, volume 16, pages 3776–3784.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *ACL*, pages 1577–1586.

Mingyue Shang, Zhenxin Fu, Nanyun Peng, Yansong Feng, Dongyan Zhao, and Rui Yan. 2018. Learning to converse with noisy data: Generation with calibration. In *IJCAI*, pages 4338–4344.

Yiping Song, Rui Yan, Xiang Li, Dongyan Zhao, and Ming Zhang. 2016. Two are better than one: An ensemble of retrieval-and generation-based dialog systems. *arXiv preprint arXiv:1610.07149*.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *NAACL*, pages 196–205.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *CVPR*, pages 2818–2826.

Chongyang Tao, Lili Mou, Dongyan Zhao, and Rui Yan. 2018. Ruber: An unsupervised method for automatic evaluation of open-domain dialog systems. In *AAAI*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 5998–6008.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *ICML*, pages 1096–1103.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. In *ICML (Deep Learning Workshop)*.

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

Jason Weston, Emily Dinan, and Alexander H Miller. 2018. Retrieve and refine: Improved sequence generation models for dialogue. *arXiv preprint arXiv:1808.04776*.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *EMNLP*, pages 2253–2263.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2018. Learning neural templates for text generation. In *EMNLP*, pages 3174–3187.

Yu Wu, Furu Wei, Shaohan Huang, Zhoujun Li, and Ming Zhou. 2019. Response generation by context-aware prototype editing. In *AAAI*.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *AAAI*, pages 3351–3357.

Jingjing Xu, Xuancheng Ren, Yi Zhang, Qi Zeng, Xiaoyan Cai, and Xu Sun. 2018. A skeleton-based model for promoting coherence among sentences in narrative story generation. In *EMNLP*, pages 4306–4315.

Tiancheng Zhao, Kyusong Lee, and Maxine Eskenazi. 2018. Unsupervised discrete sentence representation learning for interpretable neural dialog generation. In *ACL*, pages 1098–1107.

Hao Zhou, Tom Young, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. 2018. Commonsense knowledge aware conversation generation with graph attention. In *IJCAI*, pages 4623–4629.