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

Deng Cai[1] Yan Wang[2] Wei Bi[2]
Zhaopeng Tu[2] Xiaojiang Liu[2] Wai Lam[1] Shuming Shi[2]

[1]The Chinese University of Hong Kong

[2]Tencent AI Lab

NAACL, 2019

Outline

Background

Chit-chat style dialogue systems (chatbots):

- retrieval models (IR)
- Generative models (seq2seq)

Background

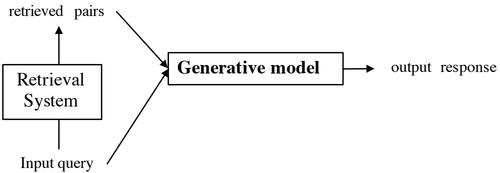
Comparison between retrieval models and generative models.

	Pros	Cons
retrieval models	informative	generalize poorly (sometimes inappropriate)
Generative models	safe	boring (e.g. “I don’t know") uninformative (repeat the query)

Existing Work

Existing retrieval-guided generative models

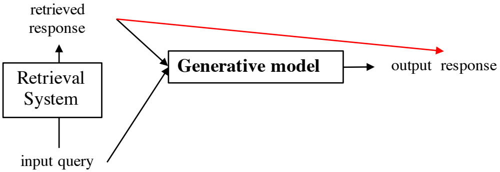
Song et al. (2016)
Weston et al. (2018)
Pandey et al. (2018)
Wu et al. (2019)



Problems

Existing retrieval-guided models are inclined to degenerate into a copy mechanism.

1. the generative models simply repeat the retrieved response without necessary modifications.
2. Sharp performance drop is caused when the retrieved response is irrelevant to the input query.



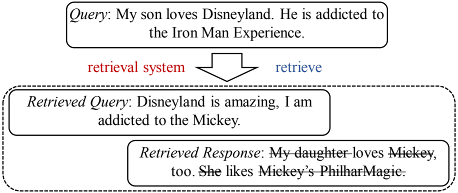
Motivation

maintain the generalization ability

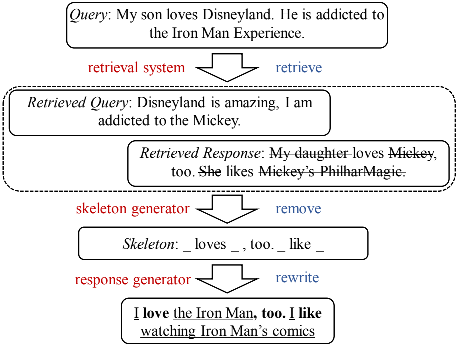
The guidance from IR results should only specify a response aspect or pattern, but leave the query-specific details to be elaborated by the generative model itself.

information filter of retrieved results

The retrieval results typically contain excessive information, such as inappropriate words or entities. It is necessary to filter out irrelevant words.

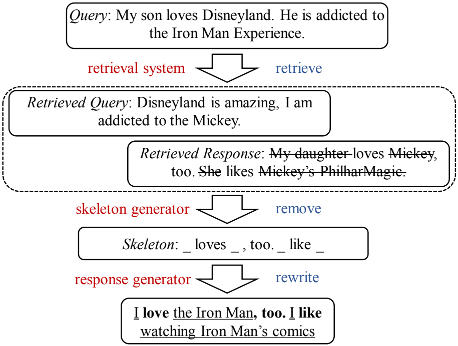


Overview



Skeleton-then-response: first constructs a response skeleton by removing some words in the retrieved response, then a response is generated via rewriting based on the skeleton.

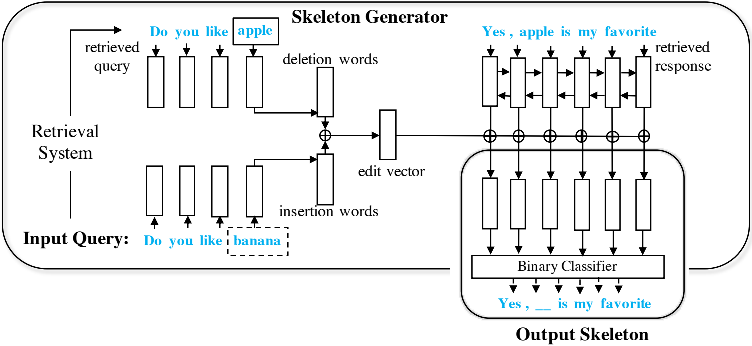
Components

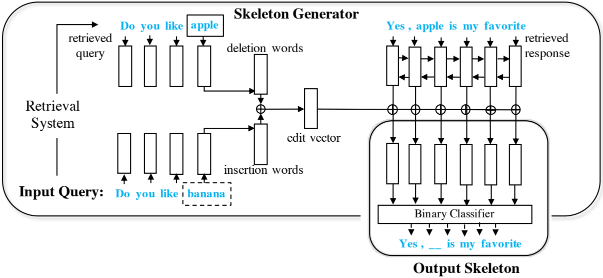


- Skeleton Generator transforms a retrieved response into a skeleton by explicitly removing inappropriate or useless information regarding the input query
- Response Generator adds query-specific details to the generated skeleton for query-to-response generation.

Skeleton Generator

The skeleton generation is formulated as a series of word-level masking actions (sequence labelling).



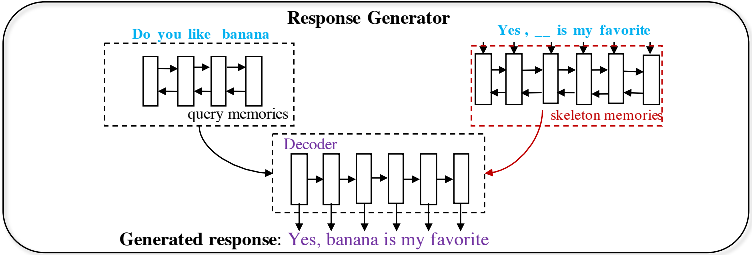


1. We compute an edit vector z based on *insertion words* I and *deletion words* D. The two bags of words highlight the changes in the dialogue context, corresponding to the changes in the response
2. The probability of masking the i-th token:

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

Response Generator

Three parts: *skeleton encoder, query encoder*, and *response decoder*.



The decoder interact with two encoders by separate attention mechanism.
The query and skeleton are fused by gated combination.

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

Integration

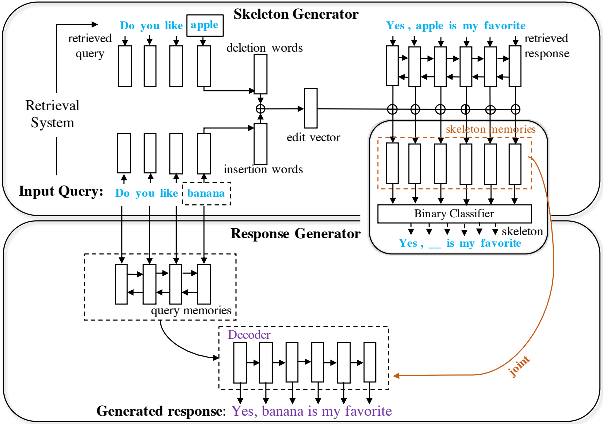
- Due to the discrete choice of skeleton words, the overall model cannot be trained end-to-end using the standard maximum likelihood estimate.
- Our solutions:
  1. Joint Integration (multi-task learning)
  2. Cascaded Integration (reinforcement learning)

Joint Integration

- We connect the skeleton generator and the response generator via a shared network architecture rather than by passing the discrete skeletons.
- The training objective is the sum of the proxy skeleton labels likelihood $L(\theta_{ske})$ and the response likelihood $L(\theta_{res})$:

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

Joint Integration



The last hidden states in our skeleton generator are directly used as the skeleton memories in response generation

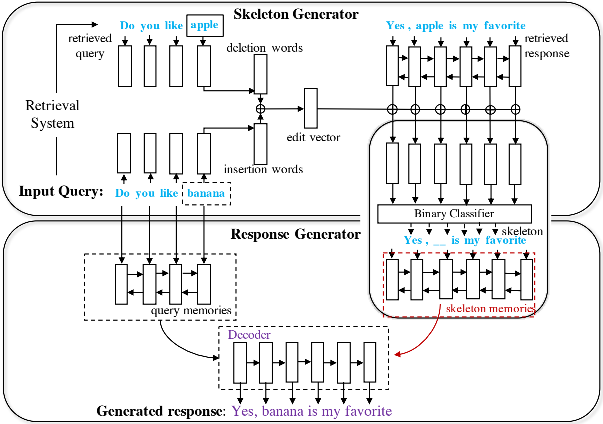
Cascaded Integration

- Policy gradient methods (Williams, 1992) can be applied to optimize the full model while keeping it running as cascaded process.
  1. first RL agent: the skeleton generator
  2. second RL agent: the response generator
- Reward design:

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

where $\hat{r}$ is the machine-generated response, r is the human-written response, and $\bar{r}$ is a random response (yet written by human).

Cascaded Integration



Setup

We use the preprocessed data in Wu et al, (2019) as our test bed.

- single-turn query-response pairs collected from Douban Group.[1]
- 5 million training quadruples (q, r, q', r') and 1000 queries for test
- It is required that $0.3 \leq Jaccard(r, r'_i) \leq 0.7$ for training quadruples.

The training quadruples for IR-augmented models are constructed based on response similarity (similar contexts may correspond to totally different responses).

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

Compared Methods

- **Seq2Seq** the standard attention-based RNN encoder-decoder model (Bahdanau et al., 2014).
- **MMI** SEQ2SEQ with Maximum Mutual Information (MMI) objective in decoding (Li et al., 2016a).
- **EditVec** the model proposed by Wu et al. (2019).
- **IR** the Lucene system is also used a benchmark.[2]
- **IR+rerank** rerank the results of **IR** by **MMI**.

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

Model Variants

- **JNT** our model with joint integration.
- **CAS** our model with cascaded integration.
- **SKP** our response generator that takes an intact retrieval response as its skeleton input (i.e., to completely skip the skeleton generation step)

Evaluation Metrics

- **Human Evaluation** Responses are rated on a five-point scale. A response should be scored
  1. 1 if it can hardly be considered a valid response.
  2. 3 if it is a valid but not informative response.
  3. 5 if it is an informative response, which can deepen the discussion of the current topic or lead to a new topic.
  4. 2 and 4 are for decision dilemmas.
- **Dist-1 & Dist-2** the number of unique uni-grams (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)

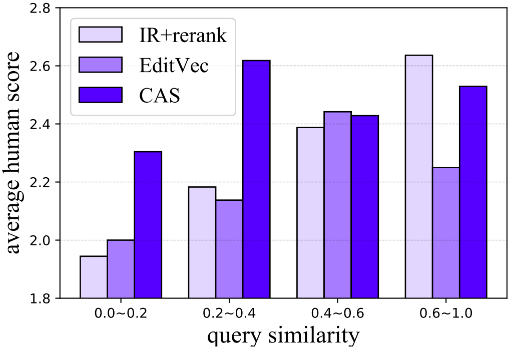
Response Generation Results

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

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 †.

Results and Analysis

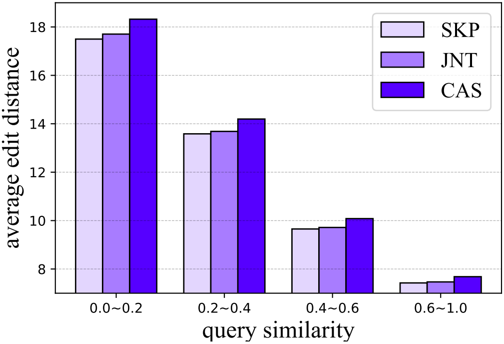
Response quality v.s. query similarity



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.

Results and Analysis

Changes between retrieved and generated responses v.s. query similarity



The use of skeletons makes the generated response deviate more from its prototype response. The changes between the generated response and the prototype response depend on the context similarity.

Results and Analysis

Single v.s. Multiple Retrieval Pair(s)

- **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 re-ranked 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.

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

Possible reason: The response generator receives many heterogeneous skeletons, yet it has no idea which to use.

Conclusion and Future Work

- The skeleton-then-response helps reduce the search space of possible responses and provides useful elements missing in the given query, resulting in more informative responses.
- It might be used for controllable dialogue response generation.
- The response skeleton could come from other sources, for example, a knowledge base.

Thanks!

thisisjcykcd@gmail.com