SCIENCE CHINA
Information Sciences

• LETTER •

December 2019, Vol. 62 229103:1–229103:3
https://doi.org/10.1007/s11432-018-9479-3

Reformulating natural language queries using sequence-to-sequence models

Xiaoyu LIU, Shunda PAN, Qi ZHANG*, Yu-Gang JIANG & Xuanjing HUANG

School of Computer Science, Fudan University, Shanghai 201203*, China*

Received 12 March 2018/Revised 7 May 2018/Accepted 5 June 2019/Published online 6 November 2019

Citation Liu X Y, Pan S D, Zhang Q, et al. Reformulating natural language queries using sequence-to-sequence models. Sci China Inf Sci, 2019, 62(12): 229103, https://doi.org/10.1007/s11432-018-9479-3

Dear editor,
In recent years, reformulating natural language queries has attracted significant amount of attention from both industry and academia. However, there is a lexical chasm between natural language queries and relevant documents. If natural language queries are input directly, the results are usually unsatisfactory. Therefore, reducing the lexical chasm between user queries and relevant documents is crucial for information retrieval systems.

This "lexical chasm" problem has attracted increasing attention from both academic and enterprise communities. Query reformulation, which attempts to alleviate vocabulary mismatch by changing a given natural language query into an alternative query, is considered as an effective solution for this problem, and various related approaches have been proposed [1–4]. For example, Jones et al. [2] proposed generating a new query to replace the natural language query and considered query reformulation as a machine translation problem, where user queries are treated as one language and the corresponding reformulated queries are treated as another language. Statistical machine translation models [1, 3] and neural machine translation models [4] have been proposed to rewrite the natural language queries. However, such models cannot effectively generate out-of-vocabulary (OOV) and low-frequency words in keyword queries. For example, some entity words (e.g., Shanghai) appear in few natural language queries and generating these words from a fixed-size vocabulary as keywords is often difficult.

In this study, we aim to address the "lexical chasm" by translating natural language queries into keyword queries [4]. Typically, extractive and abstractive methods are employed for query translation: extractive methods and abstractive methods. Extractive methods extract keywords directly from natural language queries, while abstractive methods generate keywords from a fix-sized vocabulary, which may generate novel keywords that do not appear in the source text. This study adopts the abstractive method because some query keywords are semantically similar to the natural language query but do not actually appear in the given query, We first explore a recurrent neural network (seq2seq) model with an attention mechanism [5, 6] to translate natural language queries to keyword queries.

Bidirectional LSTM encoder. Our encoder uses a single-layer bidirectional long short-term memory (LSTM) as the (RNN) unit to obtain the semantics within a sentence. The LSTM is a variant of an RNN and is used to solve the gradient vanishing problems in conventional RNNs. The LSTM introduces a gate mechanism and a memory cell to avoid gradient vanishing. As a result, LSTM network can provide better performance than a conventional RNN in most cases. Here, an LSTM unit comprises input gate $\boldsymbol{i}$, forget gate $\boldsymbol{f}$, memory cell $\boldsymbol{c}$, and an output gate $\boldsymbol{o}$, which are defined as fol-

* Corresponding author (email: qz@fudan.edu.cn)

© Science China Press and Springer-Verlag GmbH Germany, part of Springer Nature 2019

lows:

$$\begin{aligned} \boldsymbol{i}_i &= \sigma(\boldsymbol{W}_i \boldsymbol{h}_{i-1} + \boldsymbol{U}_i \boldsymbol{e}_{x_i} + \boldsymbol{b}_i), \\ \boldsymbol{f}_i &= \sigma(\boldsymbol{W}_f \boldsymbol{h}_{i-1} + \boldsymbol{U}_f \boldsymbol{e}_{x_i} + \boldsymbol{b}_f), \\ \widetilde{\boldsymbol{c}_i} &= \tanh(\boldsymbol{W}_{\widetilde{c}} \boldsymbol{h}_{i-1} + \boldsymbol{U}_{\widetilde{c}} \boldsymbol{e}_{x_i} + \boldsymbol{b}_{\widetilde{c}}), \\ \boldsymbol{c}_i &= \boldsymbol{f}_i \odot \boldsymbol{c}_{i-1} + \boldsymbol{i}_i \odot \widetilde{\boldsymbol{c}}_i, \\ \boldsymbol{o}_i &= \sigma(\boldsymbol{W}_o \boldsymbol{h}_{i-1} + \boldsymbol{U}_o \boldsymbol{e}_{x_i} + \boldsymbol{b}_o), \\ \boldsymbol{h}_i &= \boldsymbol{o}_i \odot \tanh(\boldsymbol{c}_i), \end{aligned} \tag{1}$$

where σ is the element-wise sigmoid function and $\odot$ is the element-wise product function. $\boldsymbol{e}_{x_i}$ is the embedding of word x_i, which maps discrete input word x_i to a distributed embedding vector. $\boldsymbol{h}_i$ is the hidden state vector storing all useful information of a processed sequence. $\boldsymbol{W}, \boldsymbol{U}$ and $\boldsymbol{b}$ are trainable parameters.

Unipdirectional LSTM decoder with attention mechanism. The decoder is a unidirectional LSTM with hidden states $\boldsymbol{s}$, and it generates a variable length sequence $y = (y_1, y_2, \ldots, y_n)$ through a conditional language model:

$$s_t = f(s_t, c_t), \tag{2}$$

$$p(y_t|y_{1,\ldots,t-1}, x) = g(s_t), \tag{3}$$

where f is the LSTM unit and non-linear function g is a softmax classifier that outputs the probabilities of all the words in the fixed-size vocabulary. Here, the decoder is initialized with a summarization of the entire source text $[\overrightarrow{h}_n, \overleftarrow{h}_1]$, which is given by

$$s_0 = W_s[\overrightarrow{h}_n, \overleftarrow{h}_1], \tag{4}$$

where W_s is a trainable parameter matrix, that learns to map the concatenation of forward encoder final hidden state $\overrightarrow{h}_n$ and backward encoder final hidden state $\overleftarrow{h}_1$ to the semantic spaces of the decoder. The encoder and decoder networks are trained jointly to maximize the conditional probability of the target sequence. During the decoding process, the decoder receives the word embedding of the previous word at each time step t (in the training phase, this is the previous word of the reference keyword, and at test time, it is the previous word emitted by the decoder).

Pointer model. We introduce a copying mechanism into the attentional seq2seq models to jointly copy and generate keywords during the decoding process. This model is a hybrid between the attentional seq2seq model and a pointer network. Thus this model can generate keywords from a fixed-size vocabulary and extract keywords directly from the original query. Our decoder obtains the current keyword at each time step by calculating: (1) the generation probability $p_g \in [0,1]$; (2) the copying probability $p_c \in [0,1]$; (3) the probability distribution over the vocabulary P_{vocab}; and (4) the probability distribution over the word in the source text P_x. The final vocabulary distribution over the extended vocabulary is calculated as

$$p(w) = p_g P_{\text{vocab}}(w) + p_c P_x. \tag{5}$$

Specifically, $P_{\text{vocab}}(w)$ is a probability distribution of all words in the vocabulary and is calculated as

$$\begin{aligned} P_{\text{vocab}}(w) &= \text{softmax}(g(s_t)) & (6) \\ &= \text{softmax}(W_g[s_t, c_t] + b_g), & (7) \end{aligned}$$

where W_g and b_g are trainable parameters, s_t and c_t is the hidden states and context vectors of the decoder. The generation probability p_g is calculated as follows:

$$p_g = \sigma\left(w_c^{\mathrm{T}} c_t + w_h^{\mathrm{T}} h_t + w_x^{\mathrm{T}} x_t + b_p\right), \tag{8}$$

where w_c and w_h is a vector and b_p is scalar. The copying probability is simply calculated as follows:

$$p_c = 1 - p_g. \tag{9}$$

P_x is a probability distribution of all words in the source text and is calculated as

$$P_x(w) = \sum_{i:x_i=w} \alpha_{t,i}, \tag{10}$$

if w does not appear in the source text, then $P_x(w)$ is zero. One of the main advantages of this model is that it can generate OOV words that appear in the source text.

Experiment. The dataset used in our experiments comprises a collection of triples (natural language query, keyword query, and best answer). The natural language query collected from the question of "Baidu Knows". The best answer is the answer selected by the questioner among all answers or simply the answer with the most votes. The keyword query is constructed by extracting some words from the relevant documents, and the keyword query helps us better search for relevant documents. In this context, relevant documents are documents that are similar to the best answer or contain the best answer.

We used two common metrics (Hits@K and Precision@K) to evaluate retrieval performance for original language queries. First, we translated questions into keyword queries and obtained search results using the Baidu search engine. We then used the best answer to measure the quality of the search results. We employed the edit distance ratio to determine whether the best answer was retrieved. Note that a search result is considered positive if its similarity to the best answer is greater than 0.5.

Results and discussion. To evaluate the proposed method, we compared the following methods using the constructed corpus:

Raw query. Here, the original natural language query is given to the search engine directly without modification, which can be considered as a baseline for retrieval performance.

Attentional seq2seq. This is the general encoder-decoder model with the attention mechanism, which is widely used in several NLP tasks such as machine translation [2, 3], speech recognition [1, 5], and text summarization [6].

Sequence labeling. This method is an extractive method, and it can only extract words from natural language queries as the keywords of keyword queries. Here, we consider the keyword extraction as a word-based sequence labeling task and adopt the Bi-LSTM model.

Pointer network. This method [4] is also an extractive method and can be considered as a variant of the proposed model without the generator decoder module.

ATS2S + SL. This method directly combines the results of the attentional seq2seq and sequence labeling models. Then, it filters out duplicate words to construct keyword queries.

ATS2S + PN. This method first directly combines the results of the attentional seq2seq and pointer network models, and then it filters out duplicate words to construct the keyword queries.

Joint ATS2S + SL. Here, the attentional seq2seq and sequence labeling model share the same Bi-LSTM encoder. The sequence labeling model used the Bi-LSTM encoder to extract keywords and the attentional seq2seq model generated keywords by its decoder. The two models were trained jointly together.

Table 1 lists the performances of the different methods on the dataset. The result shows that the attentional seq2seq model was slightly better at generating generative keywords, however it could not handle extractive keywords effectively. The sequence labeling model was able to better extract extractive keywords, however, it could not identify generative keywords. Note that both extractive keywords and generative keywords are important for retrieval. The proposed model could simultaneously handle extractive and generative keywords well and achieved much better recall performance for the total keywords than the other methods. Therefore, the proposed model demonstrates the best performance relative to processing natural language queries.

Table 1 Performance of different methods

Models	H@5	H@10	P@3	P@5	P@10
Raw query	25.9	29.3	11.8	8.3	6.8
Attention seq2seq	19.1	21.7	7.9	5.7	4.7
Sequence labeling	22.2	25.1	9.6	6.9	5.7
Pointer network	28.3	31.2	14.1	10.6	8.5
ATS2S + SL	29.8	34.2	13.2	10.0	8.2
ATS2S + PN	28.7	32.0	13.3	9.9	8.0
The proposed model	**37.1**	**44.1**	**16.3**	**11.5**	**9.6**

Acknowledgements This work was partially supported by National Key Research and Development Plan (Grant No. 2017YFB1002104), National Natural Science Foundation of China (Grant Nos. 61532011, 61751201, 61473092, 61472088), and STCSM (Grant No. 16JC1420401, 17JC1420200). The authors would like to thank the anonymous reviewers for their helpful comments.

References

1 Riezler S, Liu Y. Query rewriting using monolingual statistical machine translation. Comput Linguist, 2010, 36: 569–582

2 Jones R, Rey B, Madani O, et al. Generating query substitutions. In: Proceedings of the 15th International Conference on World Wide Web, Edinburgh, 2006. 387–396

3 Gao J F, He X D, Xie S S, et al. Learning lexicon models from search logs for query expansion. In: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Jeju Island, 2013. 666–676

4 Song H J, Kim A, Park S B. Translation of natural language query into keyword query using a RNN encoder-decoder. In: Proceedings of International ACM SIGIR Conference, 2017. 965–968

5 Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. 2015. ArXiv: 1409.0473v6

6 Luong M T, Pham H, Manning C D. Effective approaches to attention-based neural machine translation. 2015. ArXiv: 1508.04025

7 Gu J, Lu Z, Li H, et al. Incorporating copying mechanism in sequence-to-sequence learning. 2016. ArXiv: 1603.06393

8 Riezler S, Liu Y, Vasserman A. Translating queries into snippets for improved query expansion. In: Proceedings of International Conference on Computational Linguistics, Manchester, 2008. 737–744

9 Rush A M, Chopra S, Weston J. A neural attention model for abstractive sentence summarization. 2015. ArXiv: 1509.00685