RUCIR at TREC 2019: Conversational Assistance Track

Xiaochen Zuo
zuoxc@ruc.edu.cn
Renmin University of China
Haidian, Beijing

Xue Yang
ruc_yangx@ruc.edu.cn
Renmin University of China
Haidian, Beijing

Zhicheng Dou
douzc@ruc.edu.cn
Renmin University of China
Haidian, Beijing

Ji-Rong Wen
jrwen@ruc.edu.cn
Renmin University of China
Haidian, Beijing

ABSTRACT

In this paper we give a brief overview of the RUCIR group's participation in the TREC 2019 Conversational Assistance Track. All our runs for the Conversational Assistance Track are on the full MS MARCO Conversational Search Sessions dataset and use the online Indri retrieval system hosted at CMU.

For the Conversational Assistance Track, our runs try to solve conversational retrieval problems from two directions: One is to improve the search results by modifying the user's current query, including query reference resolution and incorporate the information from user's history queries in the same session. Run 1, Run 2 and Run 4 use this method. The other direction is to design a neural network to model user's global search intent and current search intent to get the retrieval results and run3 uses this method.

KEYWORDS

conversational search, information retrieval, reference resolution, neural network

ACM Reference Format:
Xiaochen Zuo, Xue Yang, Zhicheng Dou, and Ji-Rong Wen. 2018. RUCIR at TREC 2019: Conversational Assistance Track. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/1122445.1122456

1 CONVERSATIONAL INFORMATION RETRIEVAL

Conversational information retrieval treats the user's search behavior as a process of dialogue between the user and the search engine. But the difference with the dialogue robot is that the search engine cannot explicitly give the user an answer similar to the communication between people, but by returning the retrieved document list as an implicit response to the user, the implicit meaning here is that the search engine's response to the user is included in the document and requires the user to read and select. This kind of interaction makes the conversational retrieval and the traditional information retrieval task have significant differences, from the traditional unilaterally dominant interaction mode to the mutual interaction mode. On the one hand, the search engine gradually clarifies the user's query intention according to a series of queries provided by the user and returns a document more in line with the intention. On the other hand, the user also judges the search intention understood by the search engine according to the information fed back by the search engine, and adjusts the expression of the query according to the deviation between the understanding of the search engine and the actual intention, thereby providing a query that is more suitable for the true search intention.

Taking the query sequence in the actual session as an example, for the current query "What training is required for a PA", if you do not rely on the external knowledge base, it is difficult to understand what the PA here means. However, if you see "What is a physician assistant" and "physician assistant average salary" in the history query, it is not difficult to know that PA is "physician assistant". In this regard, the session-based retrieval model can also understand the meaning of the entity. In fact, the query of the conversational search will be more concise on the basis of this, for example, the current query is simplified to "What training is required", and even the previous query is simplified to "their average salary". At this time, only the first query has the word "physician assistant" which expresses the core search intent.Therefore, in the conversational information retrieval problem, it is particularly important to fully exploit the information of the user clicked document in the previous queries in the case where the amount of information queried by the user is limited.

On the other hand, the intent of all queries in a session-based information retrieval system does not vary much, but the query intent within the same session in a conversational information retrieval system can sometimes vary greatly. For example, the first query in a session is "What are the different types of macromolecules?", the second query is "Tell me about the characteristics of carbohydrates", and the sixth query is "Tell me about lipids". The six queries not only omit a lot of information as mentioned above, but also have a large difference from the intention of the second query. Therefore, the related information of the second query will be biased when

Woodstock '18, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

Table 1: Handcrafted features in this task.

Index	Description
1	Covered query terms number
2	Covered query terms ratio
3	Passage length
4	Minimum length of passage which covers query terms
5	Passage length normalized minimum length of passage which covers query terms
6	The first occurrence of the query term in the passage
7	The last occurrence of the query term in the passage
8	The distance between the fist and the last occurrence of the query term in the passage

applied to the sorting of the documents of the sixth query. To address this issue, a model needs to be designed to model the specific intent of the current query.

2 CONVERSATIONAL ASSISTANT TRACK

2.1 Question Definition

Similar to the session-based information retrieval, the user query of the conversational information retrieval study is also derived from the same session. Define session history $S = \{q_1, q_2, ..., q_{t-1}\}$, the corresponding user history clicked documents as:

$$D_c = \{\{d_{1i}\}_{i=1}^{n_1}, \{d_{2i}\}_{i=1}^{n_2}, ..., \{d_{t-1i}\}_{i=1}^{n_{t-1}}\}$$

n_i indicates the number of clicked documents related to the ith query. Given current query q_t and candidate document d_c, we need to calculate the score of the candidate document: $\text{Score}(q_t, d_c) = p(d_c|q_t, S, D_c)$. Then sort the list of documents based on the score and return the results to the user.

2.2 Run 1

Run 1 focuses on passage ranking based on the official rewritten queries which avoid coreference and conversational ambiguity. We design several text matching features manually and utilize learning to rank[3] framework. The features are shown in Table 1.

2.3 Run 2

Run 2 focuses on query rewriting to solve coreference and ambiguity in the conversational setting. The inputs are original query and query generated by running AllenNLP[1] coreference resolution. We extract the most important keyword in the original query and combine it with AllenNLP processed query and keywords from query of the last turn to form a new query. We input the three types of query into Solr and obtain three sets of documents(top-k ranked by BM25). We extract top-k keywords from all documents by TF-IDF and combine them with the keywords of original query. Then we input the keywords set into Indri to get the final ranked documents list. The flowchart is shown in Figure 1(a).

2.4 Run 3

Run 3 uses a neural network model based on key-value memory network[5] and attentive kernel based method. The model is divided into three parts: model user's overall query intent, model user's current query specific intent and the introduction of statistical features. The overall structure of the model is shown in Figure 2.

model user's overall query intent. The user's overall intent needs to be obtained through the history information of the session, especially the information from user clicked documents. But due to the distraction of the topic during the dialogue, not all historical clicked documents are related to the current query intent, so we use The key-value memory neural network to store historical information, wherein the historical query is used as a key and the document information is used as a value. It is thus possible to select document information that may be advantageous for understanding the current query intent by the degree of relevance of the historical query to the current query.

First of all, we need to represent the sentence in queries and documents. For query $q = \{w_1, w_2, ...w_l\}$, w_t represents the tth word in query. We use graph embedding [6] to get the representation of the word x_t, then we use bi-LSTM [2] to get the hidden vector h_t, which is the concatenation of hidden vectors from the forward LSTM and the backward LSTM model. Finally we get the representation of query q_t by attention mechanism:

$$r_q = H_q A^T = \sum_{i=1}^{l} \alpha_i h_i, H_q = [h_1, h_2, ..., h_l]$$
$$A = \text{Softmax}(v^T H_q W_a), A = [\alpha_1, \alpha_2, ..., \alpha_l]$$

Similarly, we can get the representation of all historical query $R_q = \{r_1^q, r_2^q, ...r_{t-1}^q\}$, the representation of all historical clicked documents $R_d = \{\{r_{1i}^d\}_{i=1}^{n_1}, \{r_{2i}^d\}_{i=1}^{n_2}\}, ..., \{r_{t-1i}^d\}_{i=1}^{n_{t-1}}\}$, the representation of current query r_t^q and the representation of candidate document r_d^t. All the historical clicked documents corresponding to the same historical query are averaged as the value matrix V stored in the key-value memory neural network:

$$v_i = \frac{1}{n_i} \sum_{k=1}^{n_i} r_{ik}^d, V = [v_1, v_2, ..., v_{t-1}]$$

The corresponding key value matrix K is the representation of the historical query, it means $K = R_q = \{r_1^q, r_2^q, ..., r_{l-1}^q\}$. After calculating the representation r_t^q of the current query, in order to better understand the intent of the current query through the historical knowledge through the memory neural network, historical queries that similar to the current query are selected, and the memory units with the corresponding key are read, then the memory cells are represented by weighting the memory cells as r_m:

$$r_m = V(r_t^q K)^T$$

Finally, the obtained memory vector representation interacts with the candidate document representation vector to measure the similarity between the candidate document and the user's overall query intent. The similarity is the first matching feature $f_1 = r_t^d W_1 r_m^T$

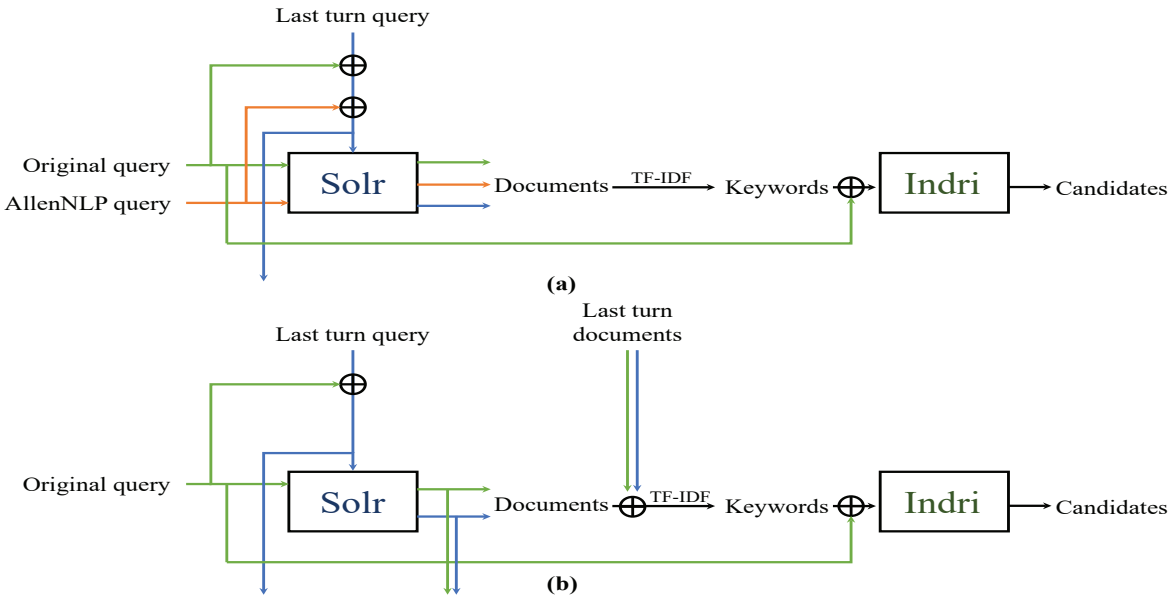

Figure 1: Two methods for coreference resolution.

model user's current query specific intent. In the conversational retrieval problem, the user's query intention in the whole session is relatively scattered. Therefore, in the process of understanding the user's current query intent, it is impossible to fully refer to the similarity with the past query process, and also needs to analyze the difference between the current query and the historical query. Some words that appear frequently in historical queries, if they appear in the current query, may reflect the user's query intent, but these words have a relatively low amount of information relative to the current query, their filtering effect of the candidate document is often not as obvious as some new words. In response to this situation, we employs a kernel-based neural network model to model the current specific query intent.

Referring to the KNRM model [7], our model interacts with the candidate document in a similar manner, but in order to emphasize the words representing the current specific intent, we introduce weight for each query word in the KNRM model, the weight a^{new} measures the freshness of the corresponding word, which is calculated as follows:

$$a_t^{\text{new}} = 1 - \max_{x_i^h \in X_h} (x_i^h W_2 x_t), a^{\text{new}} = [a^{\text{new}_1}, a^{\text{new}_2}, ..., a^{\text{new}_l}]$$

X_h indicates the representation of words that appear in the historical queries. The larger the value of a_t^{new}, the more the word x_t represents the specific intent of the current query, that is, it contains a higher amount of information. In addition, the idf of the word can also reflect the amount of information contained in the word, so we add idf to the calculation of the weight:

$$a_t^{\text{new}} = a_t^{\text{new}} * idf_t$$

In order to calculate the correlation between the current query and the candidate document, it is first necessary to interact with the word vector matrix to obtain the similarity matrix M:

$$M_{ij} = x_i^q W_3 x_j^{d^T}$$

Then, referring to the use of the kernel method in the KNRM model, multiple Gaussian kernels are used to calculate the similarity of word vectors under different distributions, and the we obtain k-dimensional matching features $\phi(M) = [K_1(M), K_2(M), ...K_k(M)]$:

$$K_k(M) = \sum_i \log(a_i^{\text{new}} F_i)$$

$$F_i = \sum_j \exp(-\frac{(M_{ij} - \mu_k)^2}{2\sigma_k^2})$$

Finally, the obtained k-dimensional matching feature is passed through the fully connected layer to obtain the similarity between the current query and the candidate document: $f_2 = \tanh(W_4 \phi(M) + b)$.

statistical features. In order to more directly measure the relationship of candidate documents to the entire query sequence, we introduce 114-dimensional statistical features. Considering that the first sentence of each document is often of a summary nature, the first 57-dimensional feature measures the relationship between the entire candidate document and the user query sequence, and the last 57-dimensional feature measures the the relationship between the first sentence of the candidate document and the user query sequence.

Among them, there are 3-dimensional feature associated with the basic statistical features of the document, 4-dimensional feature associated with the word frequency, 5-dimensional feature associated with the tf-idf value of the document, 4-dimensional feature associated with the standardized word frequency, 7-dimensional feature associated with the query word, and the vector similarity

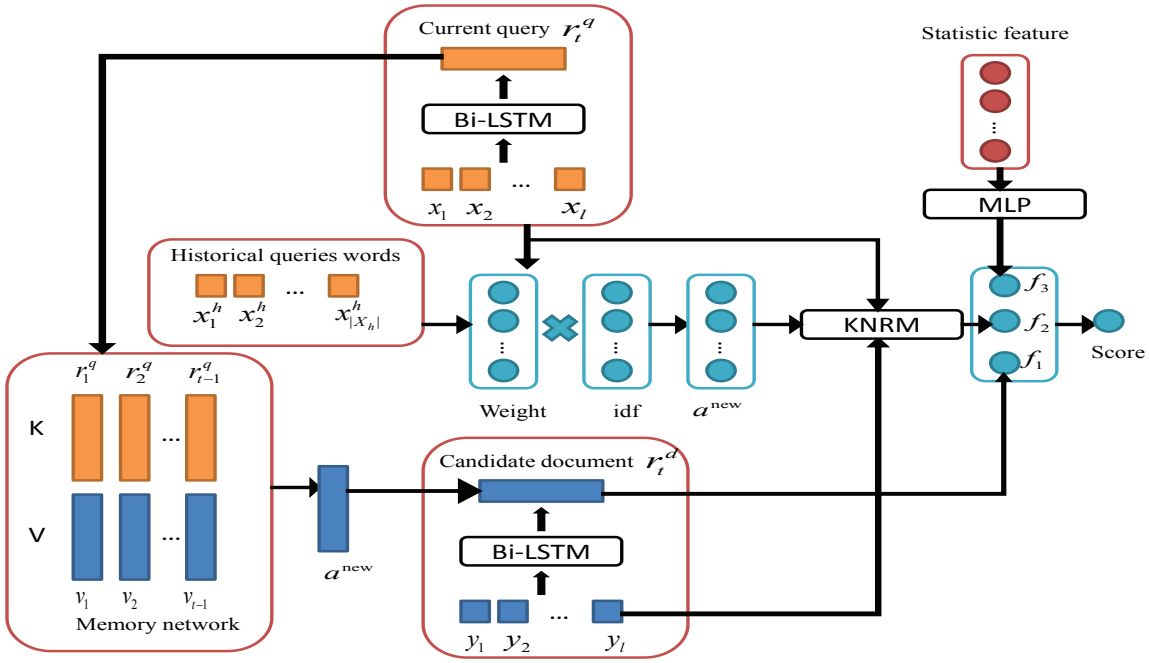

Figure 2: The Model Structure of Run 3.

feature between the document and the query word has 24 dimensions. Here, the overall word vector of the document and the query has two calculation methods, one is to directly add all the word vectors, the other is to add the word vectors by using the idf of each word as the weight. Since we use word2vec [4] and graph embedding [6] to represent word vectors, there are 4 calculation methods for each similarity, and we use 6 similarity calculation methods. Therefore, the vector similarity feature between the document and the query word has a total of 24 dimensions. Finally, the current query words are respectively connected with the previous i queries to obtain 10 recombined queries, and the 10 recombination queries respectively calculate the BM25 value and the Rouge-L value with the document to obtain 20-dimensional features. Therefore, we obtain a total of $57 \times 2 = 114$ dimensions. We pass the 114-dimensional feature through a multi-layer perceptron to obtain a 1-dimensional matching feature f_3.

document ranking. Finally, we combine the three matching features obtained before and get the final matching score through the fully connected layer:

$$\text{Score}(q_t, d_c) = \text{Leaky\_ReLU}(W_5[f_1; f_2; f_3])$$

The model training process adopts the pairwise method, that is, a pair of documents in the candidate document sequence, the positive example document is d_c^+, and the negative example document is d_c^-, and the training goal is to make the score difference of the pair of documents as large as possible. The loss function is:

$$L = \max(0, 1 - \text{Score}(q_t, d_c^+) + \text{Score}(q_t, d_c^-))$$

2.5 Run 4

Run 4 focuses on query rewriting to solve coreference and ambiguity in the conversational setting. The input is original query. We extract the most important keyword in the original query and combine it with keywords from query of the last turn to form a new query. We input the two types of query into Solr and obtain two sets of documents(top-k ranked by BM25) and combine them with documents retrieved by query of the last turn. We extract top-k keywords from all documents by TF-IDF and combine them with the keywords of original query. Then we input the keywords set into Indri to get the final ranked documents list. The flowchart is shown in Figure 1(b).

REFERENCES

[1] Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew E. Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A Deep Semantic Natural Language Processing Platform. *CoRR* abs/1803.07640 (2018). arXiv:1803.07640 http://arxiv.org/abs/1803.07640

[2] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[3] Tie-Yan Liu et al. 2009. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval* 3, 3 (2009), 225–331.

[4] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *CoRR* abs/1301.3781 (2013).

[5] Alexander H. Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-Value Memory Networks for Directly Reading Documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016.* 1400–1409. https://www.aclweb.org/anthology/D16-1147/

[6] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In *KDD*. ACM, 701–710.

[7] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-End Neural Ad-hoc Ranking with Kernel Pooling. In *SIGIR*. ACM, 55–64.