Question Answering for Technical Customer Support

Yang Li[1(✉)], Qingliang Miao[1], Ji Geng[1], Christoph Alt[2], Robert Schwarzenberg[2], Leonhard Hennig[2], Changjian Hu[1], and Feiyu Xu[1]

[1] Lenovo, Building H, No. 6, West Shangdi Road, Haidian District Beijing, China
{liyang54,miaoql1,hucj1,fxu}@lenovo.com, jgeng@uestc.edu.cn
[2] DFKI, Alt-Moabit 91c, 10559 Berlin, Germany
{christoph.alt,robert.schwarzenberg,leonhard.hennig}@dfki.de

Abstract. Human agents in technical customer support provide users with instructional answers to solve a task. Developing a technical support question answering (QA) system is challenging due to the broad variety of user intents. Moreover, user questions are noisy (for example, spelling mistakes), redundant and have various natural language expresses, which are challenges for QA system to match user queries to corresponding standard QA pair. In this work, we combine question intent categories classification and semantic matching model to filter and select correct answers from a back-end knowledge base. Using a real world user chat-log dataset with 60 intent categories, we observe that while supervised models, perform well on the individual classification tasks. For semantic matching, we add muti-info (answer and product information) into standard question and emphasize context information of user query (captured by GRU) into our model. Experiment results indicate that neural multi-perspective sentence similarity networks outperform baseline models. The precision of semantic matching model is 85%.

Keywords: Question and Answer · Answer selection
Semantic matching

1 Introduction

"My Wi-Fi is not working anymore!!!" – most mobile device users probably have faced this or similar issues in the past. Solving such questions is the task of technical customer support agents (CSAs). For frequent questions and user intents, for which solutions often exist in the form of user guides and question-answering knowledge base (QA-KB), this is a repetitive and time consuming process. For example, our QA-KB contains 242 unique standard QA pairs, and these QA pairs are assigned to 60 intents. QA system would significantly reduce the time CSAs have to invest in solving common intents, which they could then focus on more complex or previously unseen customer problems.

© Springer Nature Switzerland AG 2018
M. Zhang et al. (Eds.): NLPCC 2018, LNAI 11108, pp. 3–15, 2018.
https://doi.org/10.1007/978-3-319-99495-6_1

With the work described in this paper, we aim to automatize this task of matching instructional answers from a QA-KB to user queries. Our work combines two models, the first is question intent categories classification which for getting candidate QA pairs from QA-KB, the second is semantic matching which focuses on selecting the correct result from candidate QA pairs. Intent categories classification faces a big challenge due to the broad variety of intents in QA-KB.

User query:
Hi, I am a new moto user. I have a moto z plus. My English is bad! How do I add a google account? Can you tell me how to do it?
Standard question:
Add or remove accounts.
Answer:
You can use multiple Google Accounts and Microsoft Exchange Active Sync accounts on your device. Add account: You may need to obtain details from system administrator about the service to which the account connects. For example, you may need to know the account's domain or server address.1. Touch Apps > Settings2. Touch Accounts3. Select the type of account, e.g. Exchange, Email, Google

Fig. 1. An example of actual business data and its corresponding QA entry.

Figure 1 shows an example of real world user query and its corresponding QA entry in QA-KB. The first field is user query, the second field is standard question and the last one is answer. Table 1 lists statistic information of the data set. Through careful analysis, we obtain following characteristics. First, user queries are usually noisy (for example, spelling mistakes) and contain background or redundant information besides true intent. Second, standard questions are typically short, concise and often realized as title-like phrases, e.g. "Add or remove accounts" because this format is easy for user and CSAs to read. In addition, we can see that user query is 5 times longer than standard question so it's difficult to match these two contents. Third, answer is quite long (the average length is 78.2) and contains some information that related to user query. Due to the characteristics of the data, we add answer information into standard question. Specially, we use Gated Recurrent Unit (GRU) [1] to get context information of user query and learn long-term dependencies before multi-perspective CNN [2].

Table 1. The statistics of data.

	Num	Average length(word level)
User query	6808	31.9
Standard question	242	5.2
Answer	242	78.2

2 Related Work

Existing work on QA systems in the customer service domain has focused on answering Ubuntu OS-related questions [3], insurance domain [4] and customer relationship management [5]. Both studies show that it is in principle possible to handle longer dialogs in an unsupervised fashion and answer complex questions with the help of a noisy training set and an unstructured knowledge source. Lowe et al. [3] use a large corpus of support dialogs in the operating system domain to train an end-to-end dialog system for answering customer questions. Their results indicate that end-to-end trained systems can achieve good performance but perform poorly on dialogs that require specific domain knowledge which the model possibly never observed. In contrast, in our work we adopt a classical classification approach followed by semantically matching a user question to a set of results from a QA-KB, in order to cope with the limited amount of training data.

Most previous work on semantic matching has focused on handcrafted features. Due to the variety of word choices and inherent ambiguities in natural languages, bag-of-word approaches with simple surface-form word matching tend to poor prediction precision [6]. As a result, researchers put more emphasis on exploiting syntactic and semantic structure which are more complex and time consuming. Representative examples include methods based on deeper semantic analysis [7] and quasi-synchronous grammars [8] that match the dependency parse trees of the two sentences. Instead of focusing on the high-level semantic representation, Yih et al. turn their attention to improve the shallow semantic component, lexical semantics [9].

As development of neural network, recent work has moved away from handcrafted features and towards modeling with distributed representations and neural network architectures. Hu et al. propose two general CNN architectures ARC-I and ARC-II for matching two general sentences, and ARC-II consider the interaction between input two sentences [10]. Liu et al. propose a dual attentive neural network framework(DANN) to embed question topics and user network structures for answer selection. DANN first learns the representation of questions and answers by CNN. Then DANN learns interactions of questions and answers which is guided via user network structures and semantic matching of question topics with double attention [11]. He et al. propose a model for comparing sentences that uses a multiplicity of perspectives. They first use a CNN model to extract features at multiple levels of granularity and then use multiple similarity metrics to measure sentence similarity [2]. Feng et al. create and release an insurance domain QA corpus. The paper demonstrate 13 proposed neural network model architectures for selecting the matched answer [4]. Gaurav et al. use character n-gram embedding instead of word embedding and noisy pretraining for the task of question paraphrase identification [12]. Wu et al. propose a multi-turn sequential matching network SMN which matches two sentences in the context on multiple granularity, and distills important matching information from each pair with convolution and pooling operations. And then, a recurrent neural network (RNN) model is used to model sentence relationships [13]. These

models either calculate the similarity of users surface form question (Qu) and standard query (Q_i) in Q-A KB or calculate the similarity of Qu and answer (A_i). Our work comprehensive use Qu, Q_i and A_i. Besides we use GRU to get context information of Qu and learns long-term dependencies before CNN.

3 The Proposed Approach

3.1 Problem Formalization

The goal of our approach is to identify the Q_iA_i from n candidate QA pairs $\{(Q_1, A_1), \ldots, (Q_n, A_n)\}$ of Q-A KB that best matches Qu. Q_i is a concise and representative question such as "Connect to a Wifi network" which prototypically stands for other questions that can be answered by A_i. Figure 1 shows the example of an user query and the corresponding QA pair of QA-KB.

We hypothesize the QA pair that shares the most semantic similarity with the Qu. Following a common information retrieval approach, we use a pairwise scoring function S(Q_iA_i, Qu) to sort all candidate of the question expressed by user. Our method has two main steps, the first is intent category classification to select relevant candidate QA pairs from QA-KB. The second is semantic matching to select the best matching one from candidate QA pairs.

3.2 Intent Category Classification

Determining the correct intent category significantly reduces the number of candidate QA pairs. The dataset contains 60 intents such as "Wifi", "Screen Unlock", "Google Account", etc.

The question intent category classifier estimates the probability $p(I|Qu)$ where I denotes the intent. Our baseline approaches are Gradient Boosted Decision Trees and a linear SVM. For feature extraction, the Qu is tokenized, followed by stop-word removal and transformation into a bag-of-words representation. The classifiers use tfidf weighted unigram and bigram features. We also implement a bidirectional LSTM model [15]. In this model, each $w_i \in$ Qu is represented by an embedding $e_i \in R^d$ that we obtain from a set of pretrained distributed word representations $E = [e_1, \ldots, e_W]$. The BiLSTM output is passed to a fully-connected layer followed by a ReLU non-linearity and softmax normalization, s.t. $p(I|Qu)$ is computed as follows

$$SM(ReLU(FC(BiLSTM(E)))(Qu) \tag{1}$$

3.3 Semantic Matching

In this section, we present innovative solutions that incorporate multi-info and context information of user question into multi-perspective CNN to fulfill question paraphrase identification. The architecture of our neural network is shown in Fig. 2. The work has two same subnetworks that processing Qu and Q_iA_i

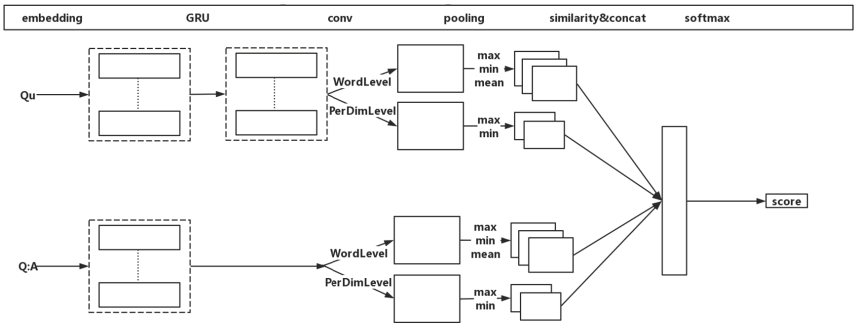

Fig. 2. Multi-perspective sentence similarity network with GRU.

in parallel after getting context by GRU. The following layer extracts features at multiple levels of granularity and uses multiple types of pooling. After that, sentence representations are compared with several granularities using multiple similarity metrics such as cosine similarity and L2 euclidean distance that are distilled into a matching vector followed by a linear projection and softmax normalization.

Multi-info. To the data, Qu is quite long, Q_i is short and contains less information. Besides, the A_i is quite long and contains some information that related to Qu. In this work, we concat Q_i and A_i of QA-KB then to compute S(Q_iA_i, Qu). User queries are always concerned with a specific product but some related standard questions for different products may be the same in the QA-KB. As you can see the example "moto z plus" in Fig. 1 which is a mobile name. Due to we do not consider the influence of different mobile, we directly replace these mobiles by the same word "Mobile". We use Product-KB and CRF algorithm to recognize the mobile in Qu. The ontology of Product-KB are constructed by senior businessmen and front-line customer service staff. Pink part of Fig. 2 indicates the structure of the Product-KB. In Product-KB, every mobile has its surface names which are mined from huge chat log. Most surface mobile name of Qu can be recognized by Product-KB.

Knowledge base hardly contains all mobiles and their corresponding surface names so we use CRF to recognize the mobile as a supplement. Features of mobile recognition are char level ngrams and word level ngrams. Maximum char level ngrams is 6 and word level ngrams is 3.

Context Multi-perspective CNN. After getting the multi-info, the input of our network are Qu and Q_iA_i. Both of them need to transfer all letters to lowercase. Given an user query Qu and a response candidate Q_iA_i, the model looks up an embedding table and represents Qu and Q_iA_i as Qu=[$e_{u,1}$,$e_{u,2}$,...,$e_{u,L}$] and Q_iA_i=[$e_{s,1}$,$e_{s,2}$,...,$e_{s,L}$] respectively, where $e_{u,j}$ and $e_{s,j} \in R^d$ are the embeddings of the j-th word of Qu and Q_iA_i respectively. L is the max length of

the two sequences. Before feed into Multi-Perspective CNN, we first employ a GRU to transform Qu to hidden vectors conM_{Qu}. Suppose that $\text{conM}_{Qu} = [h_{u,1}, h_{u,1}, \ldots, h_{u,L}]$ are the hidden vectors of Qu, then $h_{u,i}$ is defined by

$$z_i = \sigma(W_z e_{u,i} + U_z h_{u,i-1}) \tag{2}$$

$$r_i = \sigma(W_r e_{u,i} + U_r h_{u,i-1}) \tag{3}$$

$$\overline{h}_{u,i} = tanh(W_h e_{u,i} + U_h(r_i \odot h_{u,i-1})) \tag{4}$$

$$h_{u,i} = z_i \odot \overline{h}_{u,i} + (1 - z_i) \odot h_{u,i-1} \tag{5}$$

where $h_{u,0} = 0$, z_i and r_i are an update gate and a reset gate respectively, $\sigma(.)$ is a sigmoid function, and W_z, W_r, W_h, U_z, U_r, U_h are parameters. The model only gets context information of Qu and learns long-term dependencies by GRU because $Q_i A_i$ is not a sequential sentence. conM_{Qu} and $Q_i A_i$ are then processed by the same CNN subnetworks. This work applies to multi-perspective convolutional filters: word level filters and embedding level filters. Word level filters operate over sliding windows while considering the full dimensionality of the word embeddings, like typical temporal convolutional filters. The embedding level filters focus on information at a finer granularity and operate over sliding windows of each dimension of the word embeddings. Embedding level filters can find and extract information from individual dimensions, while word level filters can discover broader patterns of contextual information. We use both kinds of filters allow more information to be extracted for richer sentence modeling.

For each output vector of a convolutional filter, the model converts it to a scalar via a pooling layer. Pooling helps a convolutional model retain the most prominent and prevalent features, which is helpful for robustness across examples. One widely adopted pooling layer is max pooling, which applies a max operation over the input vector and returns the maximum value. In addition to max pooling, the model uses two other types of pooling, min and mean, to extract different aspects of the filter matches.

Multi-similarity. After the sentence models produce representations for Qu and $Q_i A_i$ then to calculate the similarity of their representations. One straight forward way to compare them is to flatten their representations into two vectors, then use standard metrics like cosine similarity. However, this may not be optimal because different regions of the flattened sentence representations are from different underlying sources. Flattening might discard useful compositional information for computing similarity. We therefore perform structured comparisons over particular regions of the sentence representations.

The model uses rules to identify local regions whose underlying components are related. These rules consider whether the local regions are: (1) from the same filter type; (2) from the convolutional filter with the same window size; (3) from the same pooling type; (4) from the same specific filter of the underlying convolution filter type. Then we use same algorithms as MPCNN [2] to calculate similarity matching vector. MPCNN use two algorithms by three similarity metrics(Cosine distance, L2 Euclidean distance, Manhattan distance) to compare

local regions. The first algorithm works on the output of holistic filters only, while the other uses the outputs of both the holistic and per-dimension filters.

4 Experiments and Discussion

We evaluate our approaches for question intent category classification, as well as semantic matching using real world user chatlog data. Next, we will introduce the dataset, QA-KB and experiment results separately.

4.1 Data Set

The dataset mainly consists of user and agent conversation records, in which user question and technical answer are stated. Each conversation record includes the full text of each utterance, chat starting and ending time, user and agent ids, and optionally a product id and an intent category assigned by the customer service agent. From 80216 user and agent conversation records, we extract 6808 user questions and annotated with a gold standard QA pair, an intent category and a product id. The distribution over the top 30 intent categories (out of 60) is shown in Fig. 4.

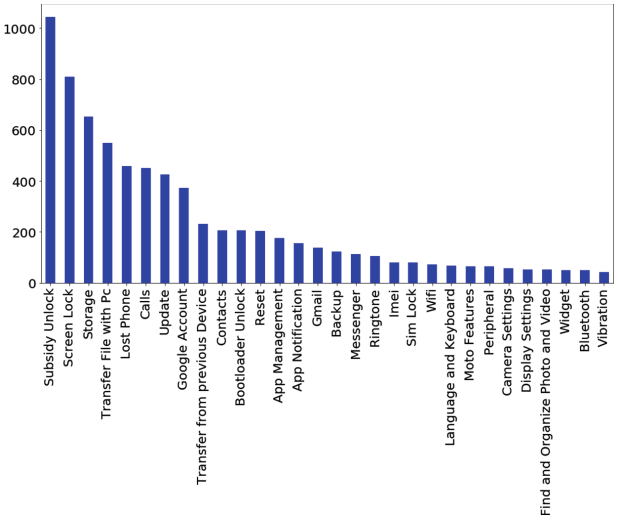

Fig. 3. Distribution of intent categories (top 30) for user question.

4.2 QA-KB and Product-KB

The KB module stores the answers of question and its relevant product. A diagram capturing the simplified structure of the KB is depicted in Fig. 4. The

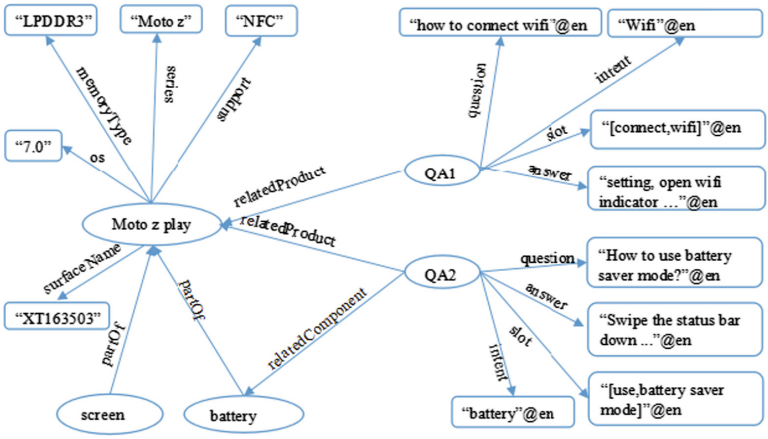

Fig. 4. Structure of the product-KB and QA-KB.

left part is the hardware and software parameters of mobile product, such as operating system, memory type, supported features, components and surface names. The right part shows QA pairs that include standard question name, corresponding answers, products, relevant slot, and an intent category. In the current version, KB includes 20 mobile products, 242 standard questions. KB totally includes more than 150000 triples.

4.3 Intent Category Classification

For question intent category classification experiments we split the dataset into 80/20 train and test sets, respectively. Hyper-parameter selection is done on the training set via 5-fold cross validation and results averaged over multiple runs are reported on the test set. For BiLSTM we use 300 dimensional GloVe word embeddings [14]. Table 2 shows the evaluation results on the dataset. The baselines perform, even outperforming the BiLSTM model.

Table 2. Intent Category Classification Results for User Question.

Model	Precision	Recall	F1
GBDT	0.67	0.68	0.67
BiLSTM	0.68	0.70	0.69
SVM	0.74	0.76	0.75

From the detailed per category results (SVM) in Table 3 we find that some categories (e.g. "Google Account and Transfer from previous Device") achieve a

Table 3. Intent category classification results for user question, Top 10 categories.

Model	Precision	Recall	F1
Subsidy unlock	0.83	0.93	0.88
Screen lock	0.84	0.92	0.88
Storage	0.79	0.85	0.82
Transfer file w. PC	0.77	0.91	0.83
Lost phone	0.87	0.91	0.89
Calls	0.79	0.87	0.83
Google account	0.64	0.72	0.68
Update	0.77	0.92	0.84
Bootloader unlock	0.93	0.67	0.78
Transfer p. device	0.67	0.74	0.70

disproportional lower performance. For example, "Google Account" is often confused with "Reset as a Google account" is generally a main topic when trying to reset a device (e.g., "Android smartphone"). It is also noteworthy that "Subsidy Unlock", "Bootloader Unlock" and "Screen Lock" are frequently confused. This is best illustrated by the example "Hi i need pin for unlock red to my moto g", which has the true category "Subsidy Unlock" but is categorized as "Screen Lock". Without knowledge about the mobile phones & contracts domain it is very difficult to understand that the customer is referring to a "pin" (subsidy unlock code) for "red" (mobile service provider) and not the actual PIN code for unlocking the phone. This example also symbolizes a common problem in customer support, where users unfamiliar with the domain are not able to describe their information need in the domain-specific terminology.

4.4 Semantic Matching

For semantic matching we evaluate TFIDF and WMD as unsupervised baselines for obtaining the most semantically similar QA pair to a given Qu. Supervised approaches include the sequential matching network (SMN), a multi-perspective CNN (MPCNN) with and without a GRU layer for user question encoding.

TFIDF and WDM. Our first baseline(TFIDF) use a tfidf weighted bag-of-words representation of Q_iA_i and Qu to estimate the semantic relatedness by cosine similarity $\cos(Q_iA_i, Qu)$.

The second baseline(WDM) leverages the semantic information of distributed word representations [16]. To this end, we replace the tokens in Q_iA_i and Qu with their respective embeddings and then compute the word mover distance [17] between the embeddings.

SMN and MPCNN. In addition to the unsupervised method we also use SMN [13] and MPCNN [2], which treats semantic matching as a classification task. The SMN first represents Q_iA_i and Qu by their respective sequence of word embeddings Ei and Ek before encoding both separately with a recurrent network, GRU [1] in this case. A word-word similarity matrix Mw and a sequence similarity matrixMs is constructed from Ei and Ek, and important matching information is distilled into a matching vector vm via a convolutional layer followed by max-pooling. vm is further projected using a fully connected layer followed by a softmax.

The MPCNN [2] first represents Q_iA_i and Qu, the following layer extracts features at multiple levels of granularity and uses multiple types of pooling. Afterwards, sentence representations are compared with several granularities using multiple similarity metrics such as cosine similarity and L2 euclidean distance. The results are distilled into a matching vector followed by a linear projection and softmax normalization.

Model Result. The description of MPCNN_GRU model is showed in Chap. 3.4. For all models except TF-IDF, we use 300 dimensional GloVe word embeddings [14]. To obtain negative samples, for each Qu, we randomly select 5 standard queries with the same intent and 5 standard queries with different intents. To alleviate the impact of unbalanced training data, we oversample positive samples. As the standard questions Q_i of most QA pairs (Q_i, A_i) are usually less then 10 tokens, we also evaluate the impact on model performance when adding the answer A_i as additional context (up to 500 characters) to Q_i. For the experimentation we randomly split the dataset 80/20 into train and test set and repeat the experiment 5 times. Hyperparameter selection is done on 10% of the training set and results are reported on the test set.

Table 4 shows the precision of each model on the semantic matching task. We see that the MPCNN and MPCNN_GRU outperform the unsupervised baseline approaches, with a 43% error reduction achieved with the MPCNN_GRU model. Intuitively it makes sense to provide the models with additional context that can be used to learn a better representation of semantic similarity. Adding a GRU to the MPCNN to encode contextual information and long-range dependencies in the user query does not really improve performance. The SMN's precision and

Table 4. Semantic matching results for user question.

Model	Without answer	With answer	Average response time
TF-IDF	0.62	0.60	null
WMD	0.60	0.58	null
SMN	0.62	0.68	200ms
MPCNN	0.72	0.84	50ms
MPCNN_GRU	0.72	0.85	55ms

recall scores are much lower than those of the MPCNN models, and only slightly higher than those of the unsupervised approaches.

Beside the precision of each semantic matching model, we also conduct experiments to evaluate the efficiency of each models. The machine configuration information in our experiment is 2 i7 CPUs with 14 cores, a memory of 125G and a disk of 930.4G. The last column of Table 4 shows the results. From Table 4, we can see MPCNN is faster than other two models. When adding GRU, the average response time increases 5ms. SMN model is slowest, because the neural network structure of SMN is more complicated than MPCNN_GRU and MPCNN. The experiment results indicate MPCNN_GRU and MPCNN is capable for real time system.

4.5 The Importance of Intent Classification for Semantic Matching

Question intent categories classification is an important step to narrow down answer candidates. In this section, we compare models with a baseline to highlight the effectiveness of intent categories classification. The baseline uses the same model as MPCNN and MPCNN_GRU, without intent categories classification so the model directly matching with all QA pairs (262) in QA-KB. Table 5 indicates that the precision of semantic matching with intent outperforms baseline models.

Table 5. Semantic matching results on baseline for User Question.

	Without intent	With intent
MPCNN	0.63	0.84
MPCNN_GRU	0.65	0.85

5 Conclusion

In this paper we presented a approach for question answering in the complex and little-explored domain of technical customer support. Our approach incorporates intent classification and semantic matching to select an answer from knowledge base. Question intent classification for a dataset with 60 intent categories and model performs reasonably well on the individual classification tasks. In semantic matching, we incorporate multi-info and context information into multi-perspective CNN to fulfill question paraphrase identification. The precision of semantic matching is 85%. Our approach outperforms baseline models. For future research, we plan to train an end-to-end model jointly add more QA pairs into QA-KB to solve more problems of customers.

References

1. Chung, J., Gulcehre, C., Cho, K.H.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555 (2014)
2. He, H., Gimpel, K., Lin, J.: Multi-perspective sentence similarity modeling with convolutional neural networks. In: 20th International Proceedings on Empirical Methods in Natural Language Processing, pp. 1576–1586. ACL, Stroudsburg (2015)
3. Lowe, R.T., Pow, N., Serban, I.V.: Training end-to-end dialogue systems with the Ubuntu dialogue corpus. Dialogue Discourse **8**(1), 31–65 (2017)
4. Feng, M., Xiang, B., Glass, M.R.: Applying deep learning to answer selection: a study and an open task. In: 3rd International Proceedings on Automatic Speech Recognition and Understanding (ASRU), pp. 813–820. IEEE, Piscataway (2015)
5. Li, X., Li, L., Gao, J.: Recurrent reinforcement learning: a hybrid approach. Computer Science (2015)
6. Bilotti, M.W., Ogilvie, P., Callan, J.: Structured retrieval for question answering. In: 30th International Proceedings on SIGIR Conference on Research and Development in Information Retrieval, pp. 351–358. ACM, New York (2007)
7. Shen, D., Lapata, M.: Using semantic roles to improve question answering. In: 12th International Proceedings on Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 12–21. ACL, Stroudsburg (2007)
8. Wang, M., Smith, N.A., Mitamura, T.: What is the Jeopardy model? A quasi-synchronous grammar for QA. In: 12th International Proceedings on Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 22–32. ACL, Stroudsburg (2007)
9. Yih, W., Chang, M.W., Meek, C.: Question answering using enhanced lexical semantic models. In: 51st International Proceedings on Association for Computational Linguistics, pp. 1744–1753. ACL, Stroudsburg (2013)
10. Hu, B., Lu, Z., Li, H.: Convolutional neural network architectures for matching natural language sentences. In: 23rd International Proceedings on Neural Information Processing Systems, pp. 2042–2050. Springer, Berlin (2014)
11. Liu, Z., Li, M., Bai, T., Yan, R., Zhang, Y.: A dual attentive neural network framework with community metadata for answer selection. In: Huang, X., Jiang, J., Zhao, D., Feng, Y., Hong, Y. (eds.) NLPCC 2017. LNCS (LNAI), vol. 10619, pp. 88–100. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-73618-1_8
12. Tomar, G.S., Duque, T.: Neural paraphrase identification of questions with noisy pretraining. In: 22th International Proceedings on Empirical Methods in Natural Language Processing, pp. 142–147. ACL, Stroudsburg (2017)
13. Wu, Y., Wu, W., Xing, C.: Sequential matching network: a new architecture for multi-turn response selection in retrieval-based Chatbots. In: 55th Annual Meeting of the Association for Computational Linguistics, pp. 496–505. ACL, Stroudsburg (2017)
14. Pennington, J., Socher, R., Manning, C.: Glove: global vectors for word representation. In: 19th International Proceedings on Empirical Methods in Natural Language Processing, pp. 1532–1543. ACL, Stroudsburg (2014)
15. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)

16. Mikolov, T., Sutskever, I., Chen, K.: Distributed representations of words and phrases and their compositionality. In: 9th International Proceedings on Advances in Neural Information Processing System, pp. 3111–3119. MIT Press, Massachusetts (2013)
17. Kusner, M., Sun, Y., Kolkin, N.: From word embeddings to document distances. In: 32nd International Proceedings on International Conference on Machine Learning, pp. 957–966. ACM, New York (2015)