Neurocomputing 316 (2018) 251–261

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Response selection with topic clues for retrieval-based chatbots

Yu Wu[a], Zhoujun Li[a,*], Wei Wu[b], Ming Zhou[b]

[a] State Key Lab of Software Development Environment, Beihang University, Beijing, China
[b] Microsoft Research Asia, Beijing, China

ARTICLE INFO

Article history:
Received 11 July 2017
Revised 11 May 2018
Accepted 27 July 2018
Available online 8 August 2018

Communicated by Dr. Y. Chang

MSC:
00-01
99-00

Keywords:
Chatbot
Deep learning
Response selection

ABSTRACT

We consider incorporating topic information into message-response matching to boost responses with rich content in retrieval-based chatbots. To this end, we propose a topic aware attentive recurrent neural network in which representations of the message and the response are enhanced by the topic information. The model first leverages the message and the response represented by recurrent neural networks (RNNs) to weight topic words given by a pre-trained LDA model and forms topic vectors as linear combinations of the topic words. It then refines the representations of the message and the response with the topic vectors through an attention mechanism. The attention mechanism weights the hidden sequences of the message and the response not only by themselves but also by their topic vectors. Thus both the parts that are important to matching and the parts that are semantically related to the topics are highlighted in the representations.Empirical studies on public data and human annotated data show that our model can significantly outperform state-of-the-art methods and rank more responses with rich content in high positions.

1. Introduction

Conversational agents are often designed to convincingly simulate how a human would behave as a conversational partner, thereby pas sing the Turing test or performing tasks or services for an individual. Building a smart conversational agents, however, is one of the most challenging tasks in natural language processing, as it requires text entailment, context understanding and language generation techniques. Recent researches on conversational agents have two dominated directions, namely task oriented dialog systems and non task oriented chatbots. The former aims to help people complete specific tasks [1], while the latter simulates humans and engages people in social conversations regarding to a wide range of issues in open domains [2].

Recently non task oriented chatbots are drawing more and more attention, and the key to building a chatbot is how to reply to a message with a proper (human-like and natural) response. Existing methods are either retrieval-based or generation-based. Retrieval-based methods [3] retrieve response candidates from a pre-built index, rank the candidates, and select a reply from the top ranked ones, while generation-based methods [4,5] leverage natural language generation (NLG) techniques to respond to a message. In this work, we study response selection for retrieval-based chatbots in a single turn scenario, because retrieval-based methods can always return fluent responses [3] and single turn is the basis of conversation in a chatbot.

The key to the success of response selection lies in accurately matching input messages with responses. In a chatbot, one can either individually use the matching scores to rank response candidates or integrate them into a learning to rank architecture. In general, matching algorithms have to overcome semantic gaps between two objects [6]. In the scenario of message-response matching, the problem becomes more serious, as proper responses in a chatbot could be diverse and contain much information. Table 1 gives an example. The response not only answers the message, but also brings in new content (e.g., the character of Batman) into the conversation. The content represents topics to talk about the movie (e.g., “character”). Such responses can facilitate the chatbot to engage its users, because they could arouse more discussions (e.g., discussions about “character of Batman”) and keep the conversation going. In practice, however, selecting such responses from others is difficult, because the extra content makes the semantic gap even bigger.

We aim to improve matching between messages and responses with rich content. Inspired by the example in Table 1, since people bring topics into responses to enrich their content, we consider introducing topic information as a kind of prior knowledge extracted from external resources into message-response matching and using the information to enhance the matching between a message and

* corresponding author.
E-mail addresses: wuyu@buaa.edu.cn, wumark@126.com (Y. Wu), lizj@buaa.edu.cn (Z. Li), wuwei@microsoft.com (W. Wu), mingzhou@microsoft.com (M. Zhou).

https://doi.org/10.1016/j.neucom.2018.07.073
0925-2312/

Table 1
A good response to a message.

Message: Is the new Batman movie worth watching?
Response: I swear you won't regret watching it. We finally get Batman as a fully rendered character. The film shows the variables he must contend with in his role as a protector of Gotham.

a response that contains rich content. We propose a topic aware attentive recurrent neural network (TAARNN) in which topic information is used to enhance representations of the message and the response. The intuition is that RNNs are good at capturing the local structure of a word sequence on both of syntactic and semantic features, but are hard to remember long-range dependencies. In contrast, latent topic models are able to capture the global semantic structure of a sentence or document but do not account for word ordering. The proposed TAARNN inherits merits of RNNs and latent topic models, so that it captures local dependencies using an RNN and global semantics of an utterance using latent topics. An informative response, a long utterance with rich contents, is difficult for simple RNNs to capture global semantic feature. In contrast, the proposed method can handle it by leveraging topic information.

Specifically, the model selects useful topic information using the massage and the response, and then leverages the distilled topic information to refine the representations of the message and the response. The matching score is calculated with the improved representations. Specifically, TAARNN first acquires topic words of the message and the response from a Twitter LDA model [7] which is pre-trained using large scale social media data outside message–response pairs. The topic words of the message hint the matching model topics that could be used in the response, and the topic words of the response indicate the model if the response and the message are in the same topics. In order to filter noise, the model assigns a weight to each topic word using a message vector and a response vector given by recurrent neural networks with gated recurrent units (GRU). Larger weights mean that the topic words are more relevant to the message or to the response and are more useful in matching. The topic words are then averaged with the weights as two vectors for the message and the response, respectively. The topic vectors then help improve the representations of the message and the response by weighting the hidden states of the GRU through an attention mechanism. The attention mechanism lets the message and the response attends to the important parts of each other with their topic vectors. Given a hidden state of the response, the attention mechanism calculates a weight for it not only by its similarity with the hidden states of the message, but also by its similarity with the topic vector of the message. The weights of the hidden states of the message are symmetrically computed. The hidden states are then linearly combined with the weights as improved representations of the message and the response respectively. In the process of representation refinement, the attention mechanism places emphasis on both the parts that are salient for recognizing the matching between the message and the response and the parts that are semantically related to the topics. Thus, topically correlated content in the message and the response (e.g., "batman" and "character" in Table 1) can be connected in matching and responses with rich content along the topics can be boosted. The message representation and the response representation are finally fed to a neural tensor network to compute their matching score. TAARNN enjoys both the power of attentive matching and the extra topic information provided by a state-of-the-art topic model. It extends the existing attention based matching model [8] by incorporating topic information into the attention process for promoting informative responses.

We conducted empirical studies on a public English data set and a human annotated Chinese data set. Evaluation results show that our model significantly outperforms state-of-the-art methods for message-response matching, and rank more responses with rich content in high positions.

Our contributions in this paper are three-folds: (1) proposal of incorporating topic information into message-response matching for chatbots. (2) proposal of a topic aware attentive recurrent neural network for matching with topics. (3) empirical verification of the effectiveness of the proposed method on public and annotated data.

2. Related work

Building a conversational agent to automatically respond to human's inputs has always been a goal of NLP researchers [9–11]. Conversational agents can be categorized into task-oriented dialogue systems and non-task oriented chatbots.

Task-oriented dialogue systems [1,12] are basically built in vertical domains and help people for certain tasks, such as booking flight, ordering food and tutoring. In dialogue systems, a user's input is parsed into some semantical representations, and the representations are exploited by a dialogue manager to determine what the response should be [13]. A dialogue manager maintains a state of the conversation by tracking who spoke last, what knowledge is private and shared, the current goal, what plan should be followed to resolve an open issue and how much of the user input was understood. After that, dialogue systems will conduct user's command and generate natural language response. The response may generated by slot fill approaches [14] or a neural network model. Together with the development of deep learning research, a trainable end-to-end dialogue system is also proposed [15], which reduces human efforts in this task.

On the other hand, chatbots, also referred to as non-task-oriented dialogue system, target at talking like a real human in open domain conversation. Research on chatbots traces back to mid-60s. ELIZA [9], a famous chatbot, relies on handcrafted templates or heuristic rules to do response generation, which requires huge efforts but can only generate limited responses. To remedy this, data driven approaches [16] have been proposed that enable chatbots to learn how to chat with human from huge available conversations history on social media and other Web 2.0 resources, such as community question and answering (cQA) or microblogging services. Data driven approaches can be categorized into retrieval based methods and generation based methods.

Retrieval based chatbots [3,16,17] select a proper response from a message-response index. A typical framework of retrieval based chatbot [3] is that a chatbot first obtains several response candidates with a search engine, and then employs a text similarity model to calculate the similarity between the message and the response candidates. There are several works that explore how to select appropriated responses for a given message. Hu et al. [6] employ a convolution network to capture the interaction between message and response. Wang et al. [17] leverage syntax features to measure the text similarity. Ji et al. [3] ensemble several text similarity features, such as cosine similarity, topic similarity, and translation score, to rank candidate responses. Generation based chatbots employ statistical machine translation techniques [18] or the sequence to sequence (S2S) framework [4,5,19–25] to generate responses. Ritter et al. [26] frame response generation as a statistical machine translation (SMT) problem. Recent progress in machine translation has inspired attempts to extend S2S to response generation. A representative S2S method is given by Shang et al. [27], which encodes message with a neural network, and generate a response with another RNN with attention mechanism. Further researches address several issues existing in the conventional S2S

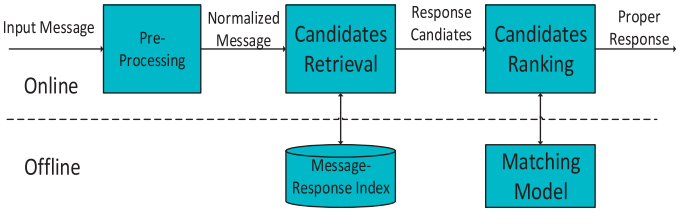


Fig. 1. The architecture of a real retrieval based chatbot, and in this paper we focus on the research of the matching model.

generation system. For example, Li et al. [20] promote the response diversity by modifying the loss function. Xing et al. [22] and Mou et al. [24] introduce extra contents to improve the informativeness of generated responses.

On top of these works, conversation history is further considered to support multi-turn conversation [28–31]. In this work, we study response selection in single turn conversation for building a retrieval-based chatbot. We propose a new message-response matching method that can incorporate topic information into matching.

Attention mechanism is an effective approach in NLP, which is firstly proposed in machine translation [32]. It assigns weights to each word in the source language when a model generates a word in target language. Recently, attention based matching methods have proven effective in tasks such as textual entailment [33] and question answering [8]. Tan et al. [8] learns the weight of each in the answer when computing the question-answer similarity. On the other hand, [33] proposes a matching model, named Match-LSTM, to learn similarity representation with two RNNs. In this work, we apply the attention technique to message-response matching in retrieval-based chatbots. Different from the existing models, our model incorporates topic information into the attention process in order to promote informative responses.

3. Retrieval based chatbots overview

In this section, we introduce a general framework of a retrieval based chatbot, as illustrated in Fig. 1, and we show the position of a matching model in the system. A retrieval based chatbot uses a repository of predefined responses and a ranking model to pick an appropriate response for a user's input. In general, a retrieval based chatbot comprises of an off-line part and an on-line part. Offline part indexes a large amount of message and response pairs to form a candidate pool. Besides, a matching model is trained in order to classifier whether or not a response is proper for a message. Online part leverages the existing index and the matching model to select a response for a user message. Specifically, it has three-stages.

- Step 1. Given a message, the system first pre-processes the message, as conversational language is informal, and even contains special words (urls, emails).
- Step 2. Retrieve response candidates from the pre-defined message-response pair index. By this mean, we can reduce the number of response candidates. As a consequence, we can apply advanced matching models to compute the similarity score between candidates and the input message. As the retrieval algorithm is not the concentration of this paper, we just use a naive search algorithm of an open source software, which will be discussed in details in experiments.
- Step 3. Rank the response candidates with a pre-train matching model. This matching model can be a single model or an ensemble model.

This paper focuses on the matching algorithm to select proper responses from candidates. We propose a neural network based method that can be used individually or as a feature in an ensemble model. In the following section, we will introduce our matching model in details and demonstrate its effectiveness based on extensive experiments.

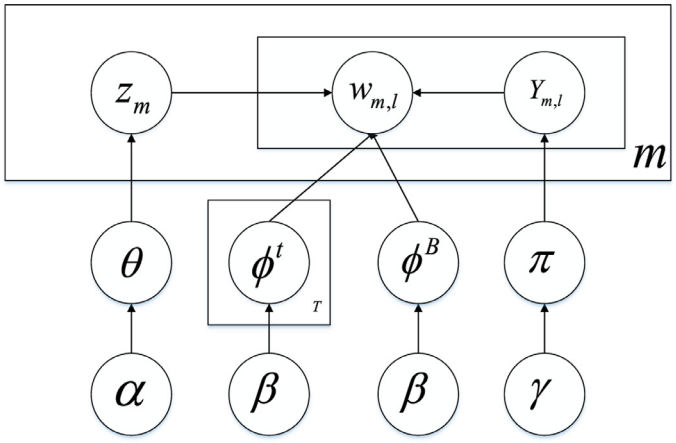


Fig. 2. Graphical model of Twitter LDA. We can infer a topic z for a document following this process.

4. Problem formalization

Suppose that we have a data set $\mathcal{D} = \{(y_i, m_i, r_i)\}_{i=1}^N$, where m_i and r_i represent an input message and a response candidate respectively, and $y_i \in \{0, 1\}$ denotes a class label. $y_i = 1$ means r_i is a proper response for m_i, otherwise $y_i = 0$. Each m_i in $\mathcal{D}$ corresponds to a topic word set $W_{m,i} = \{w_{m,i,1}, \ldots, w_{m,i,n}\}$, and each r_i in $\mathcal{D}$ has a $W_{r,i} = \{w_{r,i,1}, \ldots, w_{r,i,n}\}$ as topic words. Our goal is to learn a matching model $g(\cdot, \cdot)$ with $\mathcal{D}$ and $\{\cup_{i=1}^N W_{m,i}, \cup_{i=1}^N W_{r,i}\}$. For any message-response pair (m, r), $g(m, r)$ returns a matching score which can be utilized to rank response candidates for m.

To learn $g(\cdot, \cdot)$, we need to answer two questions: (1) how to obtain topic words, and (2) how to incorporate topic words into matching. In the following sections, we first present our method on topic word generation, then we elaborate on our matching model and learning approach.

5. Topic word generation

We employ a Twitter LDA model [7], which is the state-of-the-art topic model for short texts, to generate topic words for messages and responses. Twitter LDA first draws a mutlinomial distribution θ from a Dirichlet prior Dir(α), T multinomial distributions $\{\phi^t\}_{t=1}^T$ from Dir(β), a Bernoulli distribution π from Dir(γ), and another multinomial distribution ϕ^B from Dir(β). θ represents a topic distribution in the entire data set. $\forall t \in \{1, \ldots, T\}$, ϕ^t is a word distribution under topic t. ϕ^B is a distribution for background words. Given a message m (similar process is applied to a response r), the model then draws a topic z_m based on θ. For the lth word $w_{m,l}$ in m, an indicator $Y_{m,l}$ is first sampled from π. If $Y_{m,l} = 1$, then $w_{m,l}$ is a topic word and is sampled from ϕ^{z_m}; otherwise, $w_{m,l}$ is a background word and is sampled from ϕ^B. Fig. 2 gives the graphical model.

We estimate the parameters of Twitter LDA using a collapsed Gibbs sampling algorithm [7]. After that, we use them to assign a topic to each m_i and r_i in $\mathcal{D}$. To obtain the topic word sets, we define the salience of a word w regarding to a topic t as

$$s(w, t) = \frac{c_w^t}{c_w} \cdot c_w^t, \tag{1}$$

where c_w^t is the number of times that word w is assigned a topic t in the training data and c_w is the number of times that w is

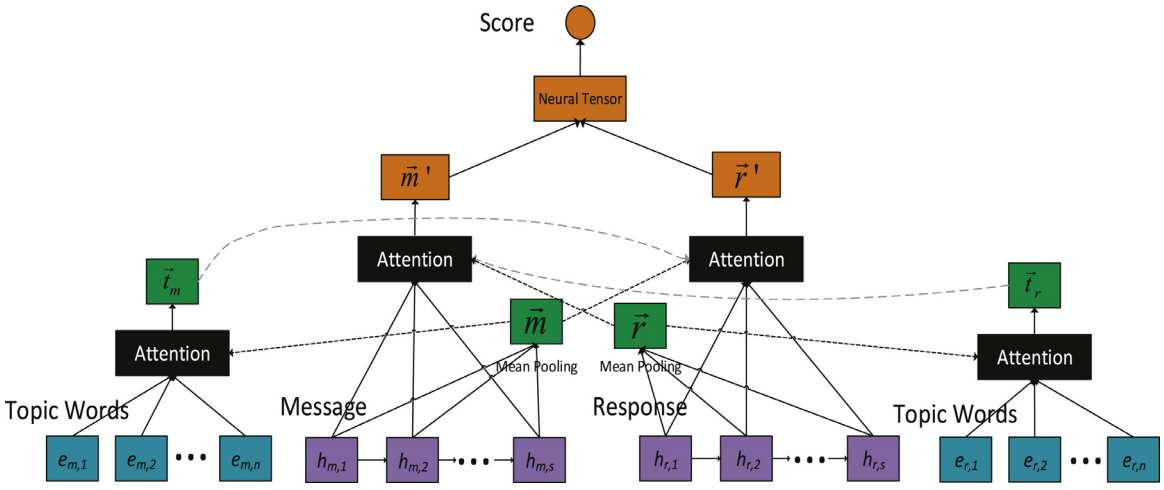


Fig. 3. The architecture of our model. The left part describes topic vector generation, while the right part illustrates the matching process.

determined as a topic word in the training data. Eq. (1) means that the salience of a word regarding to a topic is determined by the frequency of the word under the topic (i.e., c_w^t) and the probability of the word only belonging to the topic (i.e., $\frac{c_w^t}{c_w}$). $\frac{c_w^t}{c_w}$ plays a similar role to IDF in information retrieval, and is capable of reducing the importance of common words like "yes" and "cause" to topic t. With Eq. (1), we select top n words regarding to the topic of m_i and the topic of r_i to form the topic word set $W_{m,i}$ and $W_{r,i}$ respectively. In practice of a chatbot, when a new (m, r) comes, we can use the learned Twitter LDA model and follow the same technique to generate their topic words.

In our experiments, we trained Twitter LDA models using large scale questions from Yahoo! Answers and posts from Sina Weibo. The data provides topic knowledge apart from that in message-response pairs to the learning of message-response matching. The process is similar to how people learn to respond in conversation: they become aware of what can be talked about from Internet, especially from social media, and converse with others based on what they learned.

Note that in addition to LDA, one can employ other techniques such as tag recommendation [34] and keyword extraction [35] to generate topic words. One can also get topic words from other resources like wikipedia and other web documents. We leave the discussion of these extensions as our future work.

6. Topic aware attentive recurrent neural network

We propose a topic aware attentive recurrent neural network (TAARNN) to leverage the topic words from Twitter LDA in message-response matching. Fig. 3 gives the model architecture. In a nutshell, the model first leverages a message and a response represented by recurrent neural networks to attend to relevant topic words and constructs two topic vectors. It then refines the representations of the message and the response by letting them attend to the important parts of each other with their topic vectors. Different from the existing attentive models [8,33], TAARNN weights the hidden sequences of the message and the response by both themselves and their topic vectors. The advantage is that both the parts that are important to matching and the parts that are semantically related to the topics are highlighted and thus responses that contain much information related to the topics can be boosted in matching. The matching score is calculated with the refined representations.

Specifically, given a message m and a response r, our model embeds them into a vector space by a siamese neural network that consists of two recurrent neural networks with gated recurrent units (GRU) [36]. For either m or r, the model first looks up an embedding table and forms a matrix $S = [v_1, v_2, \ldots, v_s]$ where $v_j \in \mathbb{R}^d$ is the embedding of the jth word and s is the maximum sentence length. If the length of m or r does not reach s, we pad zeros after the last word. The message matrix and the response matrix are then transformed to hidden sequences by the GRU:

$$\begin{aligned} z_i &= \sigma(\mathbf{W_z} v_i + \mathbf{U_z} h_{i-1}) \\ r_i &= \sigma(\mathbf{W_r} v_i + \mathbf{U_r} h_{i-1}) \\ \widetilde{h}_i &= tanh(\mathbf{W_h} v_i + \mathbf{U_h}(r_i \odot h_{i-1})) \\ h_i &= z_i \odot \widetilde{h}_i + (1 - z_i) \odot h_{i-1}, \end{aligned} \tag{2}$$

where $h_0 = 0$, z_i and r_i are an update gate and a reset gate respectively, $\sigma(\cdot)$ is a sigmoid function, and $\mathbf{W_z}$, $\mathbf{W_h}$, $\mathbf{W_r}$, $\mathbf{U_z}$, $\mathbf{U_r}$, $\mathbf{U_h}$ are parameters. With $\mathbf{H}_m = [h_{m,1}, \ldots h_{m,s}] \in \mathbb{R}^{s \times d}$ as the hidden sequence of m, a mean pooling operation is conducted to represent m as a vector $\vec{m} = avg(h_{m,1}, \ldots h_{m,s})$. The response vector $\vec{r}$ is obtained in the same way.

TAARNN then represents the topic words given by the Twitter LDA model as two vectors, one for m and the other for r. Suppose that $W_m = \{w_{m,1}, \ldots, w_{m,n}\}$ is the topic word set of m, the model constructs a matrix $\mathbf{T}_m = \left[e_{m,1}, \ldots, e_{m,n}\right]^\top$ by looking up the embedding table (the same one with S above) where $e_{m,i} \in \mathbb{R}^d$ is the embedding of word $w_{m,i}$, and then calculates a weight vector by

$$\omega_m = \mathbf{T}_m \cdot \mathbf{A}_1 \cdot \vec{m}, \tag{3}$$

where $\mathbf{A}_1 \in \mathbb{R}^{d \times n}$ is a linear transformation learned from training data, and $\forall j$, $\omega_{m,j} \in \omega_m$ is the weight for the jth word in W_m. $\omega_{m,j}$ is further scaled to [0,1] by

$$\alpha_{m,j} = \frac{exp(\omega_{m,j})}{\sum_{p=1}^{n} exp(\omega_{m,p})}. \tag{4}$$

Topic vector $\vec{t}_m$ for message m is then defined as a linear combination of the topic words:

$$\vec{t}_m = \sum_{j=1}^{n} \alpha_{m,j} e_{m,j}. \tag{5}$$

Similarly, we have a topic vector $\vec{t}_r$ for response r. From Eqs. (3)–(5), we can see that topic words that are more relevant to the message or the response (i.e., have larger similarity with $\vec{m}$ or $\vec{r}$) will get larger weights and thus contribute more to the topic vectors. The technique here is an attention mechanism on topic words by

which we select important ones according to the message and the response.

$\vec{t}_m$ and $\vec{t}_r$ are then used to refine the representations of m and r by another attention mechanism on the hidden states of m and r. Suppose that $\mathbf{H}_r = [h_{r,1}, \dots h_{r,s}]$ is the hidden vectors of r given by the GRU, TAARNN calculates a weight vector $\omega'_r = [\omega'_{r,1}, \dots, \omega'_{r,s}]$ for $\mathbf{H}_r$. $\forall i \in \{1, \dots, s\}$, $\omega'_{r,i}$ is defined by

$$\omega'_{r,i} = tanh\left(\frac{1}{s}\sum_{j=1}^{s} h_{r,i} \cdot \mathbf{A}_2 \cdot h_{m,j} + h_{r,i} \cdot \mathbf{A}_3 \cdot \vec{t}_m\right), \tag{6}$$

where $\mathbf{A}_2$ and $\mathbf{A}_3$ are two parameters. The model then normalizes ω'_r as $\alpha'_{r,i} \in [0, 1]$ using Eq. (4) and defines the refined representation of r as

$$\vec{r}' = \sum_{j=1}^{n} \alpha'_{r,j} h_{r,j}. \tag{7}$$

In Eq. (6) the message representation and and the topic representation of the message jointly attend to important parts in the response: the weight of $h_{r,i}$ is determined by both its similarity with the hidden vectors of m (i.e., $\{h_{m,j}\}_{j=1}^{s}$) and its similarity with the topic vector of m (i.e., $\vec{t}_m$). The effect is that in Eq. (7) hidden vectors that encode information either important to recognize the matching relation of m and r or related to the topic of m will contribute more to the representation of r. The refined message representation $\vec{m}'$ is symmetrically defined in which we use $\mathbf{H}_r$ and $\vec{t}_r$ to attend to the hidden vectors in $\mathbf{H}_m$ that are either important to matching or related to the topic of r.

TAARNN finally calculates the matching score of m and r using a neural tensor network (NTN) [37]. Given $\vec{m}'$ and $\vec{r}'$, a neural tensor $s(\vec{m}', \vec{r}')$ is defined as

$$s(\vec{m}', \vec{r}') = \mathbf{f}\left(\vec{m}'^{\intercal}\mathbf{M}^{[1:h]}\vec{r}' + \mathbf{V}\left[\vec{m}'^{\intercal}, \vec{r}'^{\intercal}\right]^{\intercal} + \mathbf{b}\right), \tag{8}$$

where $f(\cdot)$ is a nonlinear function, and $\mathbf{M}^{[1:h]} \in \mathbb{R}^{n \times h \times n}$ is a tensor. The result of the bilinear tensor product $\vec{m}\mathbf{M}^{[1:h]}\vec{r}$ is a vector $\vec{v} \in \mathbb{R}^h$ with each entry a matching of m and r parameterized by a slice k of $\mathbf{M}$, $k = 1, 2, \dots, h$. $\mathbf{V} \in \mathbb{R}^{h \times 2n}$ and $\mathbf{b} \in \mathbb{R}^h$ are the other two parameters. The matching function $g(m, r)$ is defined by

$$g(m, r) = \mathbf{h}\left(\mathbf{w}^{\intercal} s(\vec{m}', \vec{r}') + \mathbf{b_2}\right), \tag{9}$$

where $\mathbf{h}$ is a softmax function, and $\mathbf{w}$ and $\mathbf{b_2}$ are parameters.

In TAARNN, topic learning and matching are conducted in two steps, because by this means we can leverage data other than message-response pairs for matching. For example, we can estimate topic words from questions in Yahoo! Answers and use them in message-response matching. These data provides extra topic information other than that in message-response pairs. TAARNN explicitly utilizes such information as prior knowledge, and that is why we call it "topic aware".

We learn $g(\cdot, \cdot)$ by minimizing cross entropy [38] with $\mathcal{D}$. Let Θ denote the parameters in our model, then the objective function $\mathcal{L}(\mathcal{D}; \Theta)$ is given by

$$-\frac{1}{N}\sum_{i=1}^{N}[y_i \log(g(m_i, r_i)) + (1 - y_i)\log(1 - g(m_i, r_i))]. \tag{10}$$

We optimize the objective function using stochastic gradient descent with Adam algorithm [39] controlling the learning rate. As regularization, we employ early-stopping [40] as it is enough to prevent over-fitting on large scale training data (1 million instances). We set the initial training rate and the batch size as 0.01 and 200 respectively.

7. Experiment

We tested our model on a public English data set and an in-house Chinese data set.

7.1. Experiment setup

The English data set is the Ubuntu Corpus [28] which consists of a large number of human-human dialogues about Ubuntu-related technique support collected from Ubuntu chat rooms. Each dialogue contains at least 3 turns, and we only kept the last turn as we focus on single-turn conversation in this work. We used the data pre-processed by Xu et al. [41],[1] in which all urls and numbers were replaced by "_url_" and "_number_" respectively to alleviate the sparsity issue. The training set contains 1 million message-response pairs with a ratio 1: 1 between positive and negative responses, and both the validation set and the test set have 0.5 million message-response pairs with a ratio 1: 9 between positive and negative responses. All the negative responses are randomly sampled rather than labeled by human annotators.

Ubuntu Corpus is a domain specific data set, and response candidates are obtained from negative sampling without human judgment. To further verify the efficacy of our model, we conducted another experiment on a Chinese data set with open domain conversations. The test set of the Chinese data set is collected following the process of a real retrieval based chatbot, and annotated by humans. Specifically, we built the Chinese data set from Baidu Tieba which is the largest Chinese forum allowing users to post and comment to others' posts. We first crawled 0.6 million text post-comment pairs as positive message-response pairs (i.e., an (m, r) with a $y = 1$). Then, for each post, we randomly sampled another comment from the 0.6 million data to create a negative message-response pair. The two sets together form a training set with 1.2 million instances. Following the same procedure, we built a validation set with 50,000 instances apart from those in training. To construct a test set, we simulated the process of a retrieval-based chatbot: we first indexed the 0.6 million post-comment pairs by an open source Lucene.Net.[2] Then, we crawled another 400 posts that are not in the training set and the validation set as test messages. For each test message, we retrieved several similar posts from the index, and collected all the responses associated with the similar posts as candidates. We recruited three human labelers to judge if a candidate is a proper response to a test message. A proper response means that the response can naturally reply to the message without any contextual information. Each candidate response received three labels and the majority of the labels was taken as the final decision. After removing messages without any proper responses (i.e., all candidates are negative ones), we obtained 328 test messages with 3418 responses. On average, each test message has 10.4 labeled responses, and the ratio between positive and negative responses is 3:2. The Chinese data set simulates the real scenario of a retrieval based chatbot that is described in Section 3, which is a complementary to the experiment on Ubuntu Corpus.

For the English data, we crawled 8 million questions (title and body) from the "Computers & Internet" category in Yahoo! Answers, and utilized these data to train the Twitter LDA model. Word embedding tables were initialized using the public word vectors[3] (trained on Twitter). For the Chinese data, we trained the Twitter LDA model and the word vectors for initializing embedding tables using 30 million posts crawled from Sina Weibo. In both data, the dimensionality of word vectors is 100.

On the Ubuntu data, we followed Lowe et al. [28] and employed recall at position k in n candidates ($R_n@k$) as evaluation metrics, while on the human annotated Chinese data, we employed mean average precision (MAP) [42], mean reciprocal rank (MRR) [43], and precision at position 1 (P@1) as evaluation metrics. Note that

[1] https://www.dropbox.com/s/2fdn26rj6h9bpvl/ubuntudata.zip?dl=0
[2] http://lucenenet.apache.org
[3] http://nlp.stanford.edu/projects/glove

$R_n@ks$ are not applicable to the Chinese data, because test messages in the data have different numbers of candidates and correct responses.

7.2. Baseline

We considered the following models as baselines:

Cosine: we calculated cosine similarity between a message and a response using their tf-idf weighted vectors. Idf is computed with the training data of two data sets, respectively.

Translation model: we learned word-to-word translation probabilities using GIZA++[4] by regarding messages in training sets as a source language and their positive responses as a target language. Following Ji et al. [3], we used translation probability $p(response|message)$ as a matching score.

Multi-layer perceptron (MLP): a message and a response were represented as vectors by averaging their word vectors. The two vectors were fed to a two-layer feedforward neural network to calculate a matching score. The first hidden layer has 100 nodes, and the second hidden layer has 2 nodes.

DeepMatch$_{topic}$: the matching model proposed in [44] which used topic information to match a message and a response. The topic information is the same as ours.

LSTM: we implemented the best model in [28] which is a matching model for retrieval-based chatbots. A message and a response are separately fed to a LSTM network and matching score is calculated with the output vectors of the LSTM networks. The difference is that in the paper, the authors also considered the context of the message, while we study single turn conversation and only considered the message.

Attention based LSTM: we implemented two state-of-the-art attention based LSTM methods for text matching, denoted as Attentive-LSTM [8] and Match-LSTM [33].

CDSSM: we selected Convolutional Deep Structured Semantic Model (CDSSM) [45] as a baseline in the experiments, which shows remarkable performance across different semantic matching tasks by modeling sentences with letter-trigrams. For Tieba dataset, we treat a Chinese character as a letter.

CNNs: the CNN models proposed by Hu et al. [6], denoted as Arc1 and Arc2 respectively, and the convolution neural tensor network [46] proposed for community question answering, denoted as CNTN.

Deep Learning to Respond: the state-of-the-art response selection model proposed by [31] for retrieval-based chatbots, denoted as DL2R. DL2R first represents sentences with LSTMs separately, and applies a CNN to extract the sentence representations. Then it calculates the matching score with a MLP.

7.3. Parameter tuning

All models were implemented with Theano, a famous deep learning framework. We tuned the parameters according to $R_2@1$ and accuracy on the dev set of Ubuntu data and Tieba data respectively. For Twitter LDA, we set $\alpha = 1/T,\ \beta = 0.01,\ \gamma = 0.01$ in the Dirichlet priors. We tuned the number of topics (i.e. T) in {20, 50, 100, 200, 500} and the maximum iteration number of Gibbs sampling in $\{100, 200, \ldots, 1000\}$. The best combination for both data sets is (200,1000). The number of topic words was tuned in $\{10, 20, \ldots, 100\}$ and set as 50 finally. The dimension of word embeddings in all neural network based models is 100. We set the dimension of hidden states (i.e., d) as 100 in all RNNs (i.e. LSTM, Match-LSTM, Attentive LSTM, DL2R, TAARNN). For Arc1, CNTN and DL2R, the CNN window size was tuned in {2, 3, 4, 5} and set as 3. The number of filters was tuned in {50, 100, 150, 200} and set as 200. For Arc2, we tuned the window size in {(2, 2), (3, 3), (4, 4)} and chose (3,3). The number of filters in Arc2 is 8. Besides we only used one layer convolution neural network, as the performance did not increase as the number of layers got larger. In CNTN and our model, we varied the number of slices in neural tensors (i.e.,h) in $\{1, 2, \ldots 10\}$ and set it as 8.

7.4. Evaluation results

Table 2 reports evaluation results on the English data and the Chinese data. We can see that on all metrics, our method performs better than baselines, and the improvement is statistically significant (t-test with p-value ≤ 0.01).

In the Ubuntu data, all models became worse when the number of candidates (i.e., n in $R_n@k$) increased and better when k increased. This is because a test message only has one positive response. Obviously, ranking the only positive one to the top becomes more difficult when more negative competitors come in, but things become easier when we allow the positive one to be ranked at lower positions.

Cosine similarity does not perform well on this task, because the semantic gap cannot be overcome by just considering lexical features. Translation model, which learns word-word translation probabilities, outperforms cosine similarity, as it considers the phenomenon that a similar meaning is expressed with the different words in language. Both DeepMatch$_{topic}$ and MLP perform badly, indicating that we cannot simply rely on topics and word embeddings to match messages and responses. Although DeepMatch$_{topic}$ uses topic information as our method, it shows much worse performance compared to TAARNN. It is mainly because DeepMatch$_{topic}$ only uses topic information and ignores word information in the matching process. Therefore, the rich information in words cannot participate in the matching process. TAARNN does not have this drawback, since it utilize $\vec{m}$, $\vec{r}$, $\vec{t}_m$ and $\vec{t}_r$ in the matching degree prediction.

Among CNN based models, Arc1 is the worst. This is consistent with the conclusions drawn by the existing works. CDSSM is better than Arc1, indicating the letter-trigram information is useful. Match-LSTM performs badly, because it is designed for textual entailment rather than response selection. Attentive LSTM is better than original LSTM demonstrates that attention mechanism is effective in message response matching. TAARNN is better than Attentive-LSTM and DL2R, indicating that the topic information is useful on message-response matching in the attention mechanism.

7.5. Comparison with variants

We first compared TAARNN with several variants to investigate if the usage of topic information in TAARNN can be replaced by simpler methods. Firstly, we expanded the message and the response with their topic words and calculated the matching score using neural tensors with the representations of the expanded message and the expanded response given by GRU as input. We denote the model as “EXP”. Secondly, we used the topic vectors as additional features in neural tensors and denote the model as “AddFea”. Thirdly, we removed the attention operation from the message and the response to the topic words and just averaged the embeddings of the topic words as topic vectors. We denote the model as “NoTLearn”. Fourthly, we only use Ubuntu and Tieba data to train the topic model instead of Yahoo Answers and Weibo data, which is denoted as “NoExtra”. It will show the contribution from the extra data. Finally, we removed the influence of the topic information from the representations of the message and the response by removing $h_{r,i} \cdot \mathbf{A}_3 \cdot \vec{t}_m$ in Eq. (6), and denote the model as “NoTopic”.

[4] http://www.statmt.org/moses/giza/GIZA++.html

Table 2
Evaluation results on the Ubuntu data and the Tieba data.

	Ubuntu data					Tieba data		
	R_2@1	R_5@1	R_{10}@1	R_{10}@2	R_{10}@5	MAP	MRR	P@1
Cosine	0.681	0.470	0.383	0.482	0.686	0.597	0.662	0.553
Translation	0.721	0.502	0.393	0.507	0.727	0.710	0.760	0.658
DeepMatch_{topic}	0.593	0.345	0.248	0.376	0.693	0.677	0.725	0.594
MLP	0.651	0.362	0.256	0.380	0.703	0.653	0.712	0.550
CDSSM	0.716	0.471	0.352	0.512	0.753	0.715	0.787	0.665
CNTN	0.743	0.489	0.349	0.512	0.797	0.731	0.797	0.670
LSTM	0.725	0.494	0.361	0.529	0.801	0.732	0.797	0.670
Arc1	0.665	0.372	0.221	0.360	0.684	0.698	0.771	0.640
Arc2	0.736	0.508	0.380	0.534	0.777	0.708	0.783	0.660
Attentive-LSTM	0.758	0.521	0.381	0.545	0.801	0.741	0.793	0.677
Match-LSTM	0.685	0.403	0.289	0.430	0.701	0.687	0.742	0.621
DL2R	0.752	0.514	0.377	0.542	0.795	0.745	0.804	0.678
TAARNN	**0.770**	**0.540**	**0.404**	**0.560**	**0.817**	**0.755**	**0.822**	**0.713**

Table 3
Evaluation results of model ablation.

Ubuntu dataset					
	R_2@1	R_5@1	R_{10}@1	R_{10}@2	R_{10}@5
EXP	0.756	0.504	0.384	0.543	0.798
AddFea	0.759	0.510	0.389	0.551	0.803
NoTLearn	0.760	0.514	0.393	0.560	0.806
NoTopic	0.758	0.504	0.389	0.550	0.798
NoExtra	0.764	0.524	0.396	0.558	0.809
TAARNN	0.770	0.540	0.404	0.560	0.817

Chinese dataset			
	MAP	MRR	P@1
EXP	0.735	0.801	0.671
AddFea	0.750	0.812	0.694
NoTLearn	0.749	0.818	0.705
NoTopic	0.750	0.808	0.685
NoExtra	0.748	0.816	0.702
TAARNN	0.755	0.822	0.713

Table 3 gives the results. "EXP" performs badly, indicating that topic information is almost useless when utilized in expansion. "AddFea" is inferior to our model, indicating that the straightforward method is not a good way to leverage topic information in chatbots. "NoTopic" is worse than "NoTLearn", meaning that even with noise, topic information is still useful in improving the representations of the message and the response. 'NoTLearn" is worse than TAARNN, indicating that attention on topic words is important as it can filter noise. TAARNN outperforms "NoExtra", showing that the improvements of TAARNN do not only come from the designed model, but also come from extra data.

7.6. Boosting informative responses

We then examined if TAARNN can boost responses with rich content in ranking. To quantify the results, we calculated the average numbers of distinct unigrams and bigrams in correct responses in top n ($n = 1, 2, 3$) results of ranking lists by Eq. (11),

$$\frac{\sum_{i=1}^{N}\sum_{j=1}^{n}\mathbb{I}(r_{i,j})\cdot \#\text{distinct ngrams}(r_{i,j})}{\sum_{i=1}^{N}\sum_{j=1}^{n}\mathbb{I}(r_{i,j})}, \tag{11}$$

where $r_{i,j}$ is the response ranked at the jth position in the ranking list of message m_i, $\mathbb{I}(r_{i,j}) = 1$ if $r_{i,j}$ is a correct one, otherwise $\mathbb{I}(r_{i,j}) = 0$, and N is the number of test messages. We denote the metrics as distinct-1 and distinct-2. Note that in [20], the authors used similar metrics to quantify the diversity of generated responses. Table 4 gives the comparison with Attentive-LSTM and DL2R on the Tieba data (note that the metrics are not applicable on the Ubuntu data because there is only one correct response and others are randomly sampled). Each entry in Table 4 shows the average number of distinct ngrams in responses, and its deviation from the average number of distinct ngrams in all correct responses. The average numbers of unigrams and bigrams in all correct responses are 9.12 and 9.63, respectively. Our model and Attentive-LSTM are better than DL2R, indicating that attention mechanism is useful to boost informative response. Furthermore, our model is superior to Attentive-LSTM as well, because we leverage topic information by a co-attention mechanism. Furthermore, top responses ranked by our model always (except top 1 distinct-1) have more distinct ngrams than the average number, demonstrating that our model is capable of boosting informative responses

Table 4
Distinct ngrams in responses.

	DL2R	Attentive-LSTM	TAARNN
Top 1 distinct-1	8.16/−0.96	8.72/−0.40	9.08/−0.04
Top 2 distinct-1	8.40/−0.72	8.96/−0.16	9.37/+0.25
Top 3 distinct-1	8.50/−0.62	9.01/−0.11	9.35/+0.23
Top 1 distinct-2	8.52/−1.11	8.75/−0.88	9.54/–0.09
Top 2 distinct-2	8.95/−0.68	9.22/−0.41	9.88 /+0.25
Top 3 distinct-2	9.18/−0.45	9.67/+0.04	9.79/+0.16

7.7. Effect of topic number

We investigate our model performance with respect to the topic number T. When $T = 1$, TAARNN degenerates to a matching model with LSTM equipped with an attention mechanism. On the other hand, $T \to \infty$ results in more information provided by the topic model can be leveraged into the neural network.

As illustrated in Fig. 4, the overall performance gets better as the topic number becomes larger in the Ubuntu Corpus. However, the performance slightly drops on the Tieba data when the topic numbers is more than 200. It indicates that the topic number is an important hyper-parameter for the matching model, and it should be tuned on different data sets in practice. Furthermore, the performance may drop if the topic number is too large.

7.8. Time complexity and space complexity

It takes more parameters and computation of TAARNN to use the topic information and attention mechanism. We compare TAARNN with simple LSTM and Attentive-LSTM in terms of time complexity and space complexity. Theoretically, the time complexity of a simple LSTM is $\mathcal{O}(n^2 \cdot s + n^2)$, where n is the embedding size and s is the sequence length, $\mathcal{O}(n^2 \cdot s)$ and $\mathcal{O}(n^2)$ are

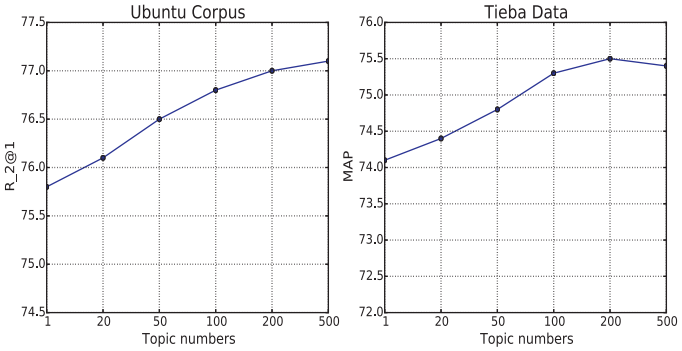


Fig. 4. Effect of topic number for two data sets. We choose $R_2@1$ as a metric for Ubuntu Corpus, while choose MAP for Tieba data set.

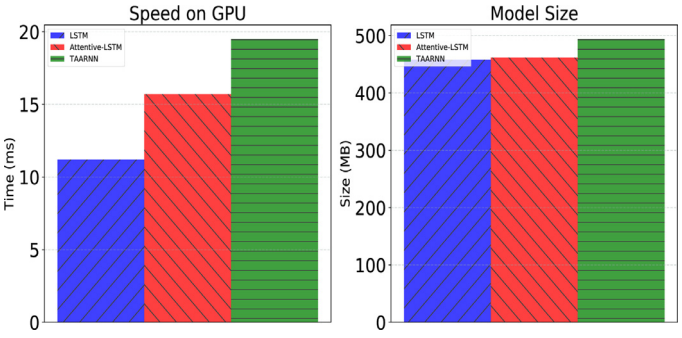


Fig. 5. Model speed and size. We compare LSTM, Attentive-LSTM and TAARNN with respect to speed and size.

the time complexities of the sentence representation and bilinear function respectively. The time complexity of Attentive-LSTM is $\mathcal{O}(n^2 \cdot s + n' \cdot n \cdot s + n^2)$, where $\mathcal{O}(n^2 \cdot s)$ is the cost on calculating sentence embeddings, $\mathcal{O}(n' \cdot n \cdot s)$ is the cost of the following attention mechanism. Here, n' is the column number of the parameter matrix in the attention mechanism, and $\mathcal{O}(n^2)$ is the time complexity of the bilinear function. TAARNN, whose time complexity is $\mathcal{O}(n^2 \cdot s + 4 \cdot n' \cdot n \cdot s + h \cdot n^2)$, is in the same order with Attentive-LSTM on time complexity. The cost on topic attention and sentence attention is $\mathcal{O}(4 \cdot n' \cdot n \cdot s)$, and the cost on neural tensor is $\mathcal{O}(h \cdot n^2)$. In practice, we tested the model efficiency using Theano on a Tesla K80 GPU with a Windows Server 2012 operation system. The left picture in Fig. 5 shows inference time of three models for one batch (200 instances). We can see that TAARNN does take more time to predict the relevance of a message-response pair, but the time cost is acceptable with a GPU.

In terms of the space complexity, TAARNN requires extra space to store topic embeddings and parameters of topic attention. TAARNN does not take up too much extra storage resources compared to baselines, because the space consumed by the increased parameters is much smaller than the space of word embedding matrix. All of the three models has the word embedding matrix, which is the bottleneck of space complexity. We also test the space expenses of these models. As shown in Fig. 5, model size of TAARNN only increases 40MB compared to a naive LSTM.

7.9. Retrieval vs. generation

We finally compared TAARNN with TA-Seq2Seq which represents the state-of-the-art generation model using topic information to improve response diversity. Following [22], we learned a TA-Seq2Seq, denoted as $\mathcal{G}$, using $\mathcal{D}$ on the Tieba data, and conducted a side-by-side human comparison on the top 1 responses of the two models on the test set (totally 400 messages). Specifically, given a message m, the generation model $\mathcal{G}$ returns the best response based on the score given by the beam search algorithm, while TAARNN ranks the candidate responses which were retrieved from the index and returns the response with the highest matching score. We recruited a native speaker to annotate which response is better.

Table 5
Retrieval v.s. generation.

	Win	Tie	Loss
TAARNN	140	209	51

The result is shown in Table 5, that TAARNN wins on 140 examples, loses on 51 examples, and on the remaining 209 examples, the judge cannot tell which one is better. The results indicate that a retrieval based chatbot with TAARNN can provide better experience to users than the state-of-the-art generation model in practice. Furthermore, according to the side-by-side comparison, we find that retrieval based chatbot is capable of generating more informative and fluent than the generation model. This characteristic may engage users in a real conversation environment.

7.10. Qualitative evaluation

We investigate why TAARNN can outperform baseline methods on Chinese data sets with examples. We translate Chinese examples to English here and show the score given by baselines and TAARNN.

Table 6(a) gives an example to show the benefit from topical keywords by a comparison between LSTM and TAARNN. The original message “Has anyone watched Christmas in August? Is it worth watching?” is relevant to the response, since the response is a famous quote in this movie. LSTM made a mistake on the relevance between the message and response, because it seems that the response talks about love, while message talks about a movie. Unlike LSTM, TAARNN first detected topical keywords of the message. Top three keywords are: movie, touching and love, and the attention weight of each word has been shown in the table. Since “love” and “touching” have high weights, TAARNN was able to recognize the relevance between the response and the message. This example demonstrates that topical keywords can help capture the topical information in a message, thus, the matching model is able to match the message-response pairs beyond literal content.

Table 6(b) gives an example to show the effectiveness of our topic attention mechanism by a comparison between TAARNN-Addfea and TAARNN. TAARNN-Addfea is a variant of TAARNN, that uses the topic vectors as additional which has been described in Section 7.5. The message “I want to watch a FANTASTIC horror movie right now!” is irrelevant to the response, as laughing is not an appropriate word to describe a horror movie. TAARNN-Addfea assigned a high score to the response, because the message and the response are similar on average word embedding, but without weights, topical keywords like “awesome” and “movie” reduced the importance of “horror movie” and brought noise to matching. On one hand, TAARNN determined a weight for each topical keyword in the message by considering the message vector, thus, TAARNN assigned a high weight to the topical keyword “horror movie”. On the other hand, a weight for a topical keyword in the response is computed using the responses vector, so as to assign a high weight to the word “comedy”. Because TAARNN could recognize the differences of the message-response on the topic level, it gives a low score to the pair. To sum up, compared with TAARNN-Addfea, TAARNN could learn a better topic representation, and demoted the response that does not talk about “horror movie”.

Table 6
Case Study for TAARNN. Since space is limited in the table, we only visualize top 3 topic words and their corresponding weights.

Effectiveness of incorporating related topic words	
Message	Has anyone watched Christmas in August? Is it worth watching?
Message topic words and weights	Movie: 0.781, TV show: 0.087, comic:0.057
Response topic words and weights	Movie: 0.499, love: 0.166, touching: 0.048
Response	Yep. I love the quote! I always knew that feelings of love would fade just like old photographs. But for you, you will always remain in my heart, as you are in my last moment.
LSTM score	0.345
TAARNN score	0.842
Label	Relevant
Effectiveness of the attention mechanism	
Message	I want to watch a FANTASTIC horror movie right now!
Message topic words and weights	Horror movie: 0.731, awesome: 0.083, movie: 0.072
Response topic words and weights	Comedy: 0.642, awesome: 0.145, fun: 0.121
Response	It is fantastic. I could not stop laughing throughout the whole movie!
TAARNN-Addfea score	0.814
TAARNN score	0.341
Label	Irrelevant

Table 7
Different responses given by LSTM, TAARNN and the generation model.

Message	Has anyone watched Christmas in August? Is it worth watching?
LSTM	Korean movies are full of warmth
Generation Model	Yes. This movie is moving.
TAARNN	Yes. I love the quote! I always knew that feelings of love would fade just like old photographs. But for you, you will always remain in my heart, as you are in my last moment.

We show responses given by the generation model, LSTM, and TAARNN in Table 7. It is shown that the generation model, TA-Seq2Seq, gives a relevant but boring response without introducing any extra information. The LSTM ranks a response, "Korean movies are full of warmth", at the top position, which introduces a discussion about Korean movie but does not provide insights to the user mentioned movie. TAARNN ranks a quote in that movie at the top position, that may engage users in the conversation environment. From this example, we can see that TAARNN is good at boosting informative responses that makes human-machine dialogue more interesting.

8. Error analysis and future work

Although TAARNN outperforms baseline methods, as well as state-of-the-art generation model. It still suffers from several problems that cannot be handled perfectly.

(1) Logical consistency. TAARNN models the message and response on the semantic level, but pays little attention to the logical consistency. This leads to several DSATs in the Tieba data set. For example, given a message "Which movie did you watch yesterday?", TAARNN ranks a response "You did not watch movie yesterday." at a high position. This model cannot realize that the word "I" should be used in the response rather than the word "you". The response is not a proper one for the message on human logic, but they are similar on semantic space. In the future, we will explore how to model logic in conversation.

(2) No correct candidates after retrieval. In the experiment, we prepared 400 messages for test, but there are only 328 messages have correct candidates after the candidate response retrieval. This indicates that there are even 72 messages have no correct candidates, which may hurt user experience in practice. Although how to retrieve candidates is not the concentration of this paper, it is still important for a retrieval chatbot. A perfect matching model still returns a bad response, if there are no candidate is relevant. In the future, we will consider improving candidate retrieval

(3) Context constraint. This paper is concerned with response selection in a single turn conversation scenario, which totally ignores context. Thus, some messages may be misunderstood if we do not consider context. For example, in the Ubuntu Corpus, a message is "seems fine.", which cannot be understood clearly if we do not consider context. If the context is given, such as a solution for a hardware problem, then responses should follow this topic. We leave response selection with context information in our future work.

9. Conclusion

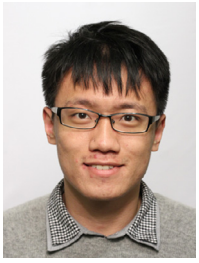
This paper proposes a TAARNN model that can leverage topic information into message-response matching for retrieval-based chatbots. TAARNN exploits topic words given by a Twitter LDA model to construct a topic vector for messages and responses respectively at the first step, then TAARNN utilizes the attention mechanism to leverage the topic vector into matching process in order to boost informative responses. We conducted experiments on two data sets, and the experimental results show that our model significantly outperforms state-of-the-art matching models on these data sets. We further investigated the informativeness of responses given by TAARNN, which outperforms baselines with a large margin. Besides, TAARNN is superior to state-of-the-art generation model according to our side by side experiment. These results demonstrate TAARNN is an effective model in retrieval based chatbot. In the future, we will investigate on the logic coherence, candidate retrieval and context constraints of retrieval based chatbots.

Acknowledgment

Yu Wu is supported by Microsoft Fellowship Scholarship and AdeptMind Scholarship.This work is supported by the National Natural Science Foundation of China (Grand Nos. 61672081, U1636211, 61370126), Beijing Advanced Innovation Center for Imaging Technology (No.BAICIT-2016001), Disaster Research Foundation of PICC.

References

[1] S. Young, M. Gašić, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, K. Yu, The hidden information state model: a practical framework for POMDP-based spoken dialogue management, Comput. Speech Lang. 24 (2) (2010) 150–174.
[2] D. Perez-Marin, Conversational Agents and Natural Language Interaction: Techniques and Effective Practices: Techniques and Effective Practices, IGI Global, 2011.
[3] J. Zongcheng, L. Zhengdong, L. Hang, An information retrieval approach to short text conversation, CoRR (2014). abs/1408.6988
[4] L. Shang, Z. Lu, H. Li, Neural responding machine for short-text conversation, in: Long Papers, 1, ACL, Beijing, China, 2015, pp. 1577–1586. July 26–31.
[5] O. Vinyals, Q.V. Le, A neural conversational model, CoRR (2015). abs/1506.05869
[6] H. Baotian, L. Zhengdong, L. Hang, C. Qingcai, Convolutional neural network architectures for matching natural language sentences, in: Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada, 2014, pp. 2042–2050.
[7] W.X. Zhao, J. Jiang, J. Weng, J. He, E.-P. Lim, H. Yan, X. Li, Comparing twitter and traditional media using topic models, in: Advances in Information Retrieval, Springer, 2011, pp. 338–349.
[8] M. Tan, B. Xiang, B. Zhou, Lstm-based deep learning models for non-factoid answer selection, CoRR (2015). abs/1511.04108
[9] J. Weizenbaum, Eliza? A computer program for the study of natural language communication between man and machine, Commun. ACM 9 (1) (1966) 36–45.
[10] S.P. Singh, M.J. Kearns, D.J. Litman, M.A. Walker, Reinforcement learning for spoken dialogue systems, in: Proceedings of the NIPS, 1999, pp. 956–962.
[11] S. Shaikh, T. Strzalkowski, S. Taylor, N. Webb, VCA: an experiment with a multiparty virtual chat agent, in: Proceedings of the Workshop on Cosmpanionable Dialogue Systems, Association for Computational Linguistics, 2010, pp. 43–48.
[12] S. Young, M. Gasic, B. Thomson, J.D. Williams, POMDP-based statistical spoken dialog systems: a review, Proc. IEEE 101 (5) (2013) 1160–1179.
[13] N. Roy, J. Pineau, S. Thrun, Spoken dialogue management using probabilistic reasoning, in: Proceedings of the Thirty-Eight Annual Meeting on Association for Computational Linguistics, Association for Computational Linguistics, 2000, pp. 93–100.
[14] G. Mesnil, Y. Dauphin, K. Yao, Y. Bengio, L. Deng, D. Hakkani-Tur, X. He, L. Heck, G. Tur, D. Yu, et al., Using recurrent neural networks for slot filling in spoken language understanding, IEEE/ACM Trans. Audio Speech Lang. Process. (TASLP) 23 (3) (2015) 530–539.
[15] T.-H. Wen, M. Gasic, N. Mrksic, L.M. Rojas-Barahona, P.-H. Su, S. Ultes, D. Vandyke, S.J. Young, A network-based end-to-end trainable task-oriented dialogue system, CoRR (2016). abs/1604.04562
[16] R. Higashinaka, K. Imamura, T. Meguro, C. Miyazaki, N. Kobayashi, H. Sugiyama, T. Hirano, T. Makino, Y. Matsuo, Towards an open-domain conversational system fully based on natural language processing, in: Proceedings of the COLING, 2014, pp. 928–939.
[17] M. Wang, Z. Lu, H. Li, Q. Liu, Syntax-based deep matching of short texts, in: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015.
[18] A. Ritter, C. Cherry, W.B. Dolan, Data-driven response generation in social media, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2011, pp. 583–593.
[19] I.V. Serban, A. Sordoni, Y. Bengio, A.C. Courville, J. Pineau, Building end-to-end dialogue systems using generative hierarchical neural network models, in: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 2016, pp. 3776–3784. February 12–17.
[20] L. Jiwei, G. Michel, B. Chris, G. Jianfeng, D. Bill, A diversity-promoting objective function for neural conversation models, in: NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016, 2016a, pp. 110–119.
[21] L. Jiwei, G. Michel, B. Chris, P.S. Georgios, G. Jianfeng, B.D. William, A persona-based neural conversation model, Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers, 2016b.
[22] C. Xing, W. Wu, Y. Wu, J. Liu, Y. Huang, M. Zhou, W.-Y. Ma, Topic aware neural response generation, in: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA., 2017, pp. 3351–3357.
[23] I.V. Serban, T. Klinger, G. Tesauro, K. Talamadupula, B. Zhou, Y. Bengio, A. Courville, Multiresolution recurrent neural networks: An application to dialogue response generation, arXiv preprint arXiv:abs/1412.6980 (2016)
[24] L. Mou, Y. Song, R. Yan, G. Li, L. Zhang, Z. Jin, Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation, COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan, 2016, pp. 3349–3358.
[25] W. Zhang, T. Liu, Y. Wang, Q. Zhu, Neural personalized response generation as domain adaptation, CoRR (2017). abs/1701.02073
[26] A. Ritter, C. Cherry, B. Dolan, Unsupervised modeling of twitter conversations, in: Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 2-4, 2010, Los Angeles, California, USA, 2010, pp. 172–180.
[27] L. Shang, T. Sakai, Z. Lu, H. Li, R. Higashinaka, Y. Miyao, Overview of the NTCIR-12 short text conversation task, in: Proceedings of the NTCIR-12, 2016, pp. 473–484.
[28] R. Lowe, N. Pow, I. Serban, J. Pineau, The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems, in: Proceedings of the SIGDIAL 2015 Conference, The 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 2-4 September 2015, Prague, Czech Republic, 2015, pp. 285–294.
[29] A. Sordoni, M. Galley, M. Auli, C. Brockett, Y. Ji, M. Mitchell, J.-Y. Nie, J. Gao, B. Dolan, A neural network approach to context-sensitive generation of conversational responses, in: NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015, 2015, pp. 196–205.
[30] X. Zhou, D. Dong, H. Wu, S. Zhao, R. Yan, D. Yu, X. Liu, H. Tian, Multi-view response selection for human-computer conversation, in: Proceedings of the EMNLP 16, 2016.
[31] R. Yan, Y. Song, H. Wu, Learning to respond with deep neural networks for retrieval-based human-computer conversation system, in: Proceedings of the Thirty-Ninth International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, 2016, pp. 55–64, doi:10.1145/2911451.2911542. July 17–21.
[32] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, CoRR (2014). abs/1409.0473 1409.0473
[33] S. Wang, J. Jiang, Learning natural language inference with LSTM, arXiv preprint arXiv:1512.08849 (2015).
[34] Y. Wu, W. Wu, Z. Li, M. Zhou, Improving recommendation of tail tags for questions in community question answering, in: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, 2016.
[35] Y. Wu, W. Wu, Z. Li, M. Zhou, Mining query subtopics from questions in community question answering, in: Proceedings of the AAAI, 2015, pp. 339–345.
[36] J. Chung, c. Gülçehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, CoRR (2014). abs/1412.3555
[37] S. Richard, C. Danqi, D.M. Christopher, Y.N. Andrew, Reasoning with neural tensor networks for knowledge base completion, in: Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States, 2013, pp. 926–934.
[38] E. Levin, M. Fleisher, Accelerated learning in layered neural networks, Complex Syst. 2 (1988) 625–640.
[39] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, CoRR (2014). abs/1412.6980
[40] S. Lawrence, C.L. Giles, Overfitting and neural networks: conjugate gradient and backpropagation, in: Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks, IJCNN, 1, IEEE, 2000, pp. 114–119.
[41] Z. Xu, B. Liu, B. Wang, C. Sun, X. Wang, Incorporating loose-structured knowledge into LSTM with recall gate for conversation modeling, CoRR (2016). abs/1605.05110
[42] R. Baeza-Yates, B. Ribeiro-Neto, et al., Modern Information Retrieval, 463, ACM Press, New York, 1999.
[43] E.M. Voorhees, et al., The TREC-8 question answering track report, in: Proceedings of the TREC, 99, 1999, pp. 77–82.
[44] Z. Lu, H. Li, A deep architecture for matching short texts, in: Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States, 2013, pp. 1367–1375.
[45] Y. Shen, X. He, J. Gao, L. Deng, G. Mesnil, A latent semantic model with convolutional-pooling structure for information retrieval, in: Proceedings of the Twenty-Third ACM International Conference on Conference on Information and Knowledge Management, ACM, 2014, pp. 101–110.
[46] X. Qiu, X. Huang, Convolutional neural tensor network architecture for community-based question answering, in: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI), 2015, pp. 1305–1311.



Yu Wu is a Ph.D. student at Beihang University. He received his B.S. degree from School of Advanced Engineering of Beihang University in 2014. His research interests include natural language processing, conversational agents, and deep neural network.

Prof. Zhoujun Li is working as a professor of Beihang University. He graduated with B.S. degree in Computer Science from Wuhan University in 1984. He received his M.S. and Ph.D. degrees in Computer Science from National University of Defense Technology in 1986 and 1999 respectively. His research interests include data mining, artificial intelligence, and information security.

Wei Wu obtained a B.S. in Applied Mathematics from Peking University in 2007 and earned his Ph.D. in Applied Mathematics from Peking University in 2012. He is a lead researcher of Microsoft Research Asia (MSRA). He joined MSRA in 2012 and have been a member of Natural Language Computing (NLC) group since then.

Prof. Ming Zhou is a Principal Researcher in Natural Language Computing group at Microsoft Research Asia. He works in the areas of machine translation and natural language processing, who has led various projects on NLP, machine translation, text mining and social network. He has authored or co-authored about 100 papers published at top conferences, and has served as area chairs of ACL, UCAI, AAAI, EMNLP, COLING, SIGIR, UCNLP for many times.