Joint Learning of Response Ranking and Next Utterance Suggestion in Human-Computer Conversation System

Rui Yan[†,♮]
[†]Institute of Computer Science and Technology
Peking University
Beijing 100871, China
ruiyan@pku.edu.cn

Dongyan Zhao[†,♮]
[†]Institute of Computer Science and Technology
Peking University
Beijing 100871, China
zhaody@pku.edu.cn

Weinan E[♮]
[♮]Beijing Institute of Big Data Research
Peking University
Beijing 100871, China
weinan@math.pku.edu.cn

ABSTRACT

Conversation systems are of growing importance since they enable an easy interaction interface between humans and computers: using natural languages. To build a conversation system with adequate intelligence is challenging, and requires abundant resources including an acquisition of big data and interdisciplinary techniques, such as information retrieval and natural language processing. Along with the prosperity of Web 2.0, the massive data available greatly facilitate data-driven methods such as deep learning for human-computer conversation systems. Owing to the diversity of Web resources, a retrieval-based conversation system will come up with at least some results from the immense repository for any user inputs. Given a human issued message, i.e., query, a traditional conversation system would provide a response after adequate training and learning of how to respond. In this paper, we propose a new task for conversation systems: joint learning of response ranking featured with next utterance suggestion. We assume that the new conversation mode is more proactive and keeps user engaging. We examine the assumption in experiments. Besides, to address the joint learning task, we propose a novel Dual-LSTM Chain Model to couple response ranking and next utterance suggestion simultaneously. From the experimental results, we demonstrate the usefulness of the proposed task and the effectiveness of the proposed model.

CCS CONCEPTS

•**Information systems** → **Retrieval models and ranking; Web applications;** *Users and interactive retrieval;*

KEYWORDS

Response ranking; next utterance suggestion; neural networks; joint learning; conversation system

SIGIR'17, August 07-11, 2017, Shinjuku, Tokyo, Japan

DOI: http://dx.doi.org/10.1145/3077136.3080843

1 INTRODUCTION

People talk to each other every day. Generally speaking, having conversations in natural languages is one of the basic ways to communicate. Since computers become a powerful tool and closely connected partner to humans, people expect that we give orders and have conversations with computers using natural languages. Such a communication interface is simple, easy, straightforward, and more importantly, human-like. For years, human-computer conversation systems have gained much attention for their functional, social, and entertainment roles in real-world applications.

To launch a human-computer conversation system is inevitably difficult and challenging. It is a hardcore problem which involves interdisciplinary techniques such as information retrieval, natural language processing, and advanced artificial intelligence. The task requires comprehensive semantics analysis to understand human utterances. The system needs to respond accordingly, either by organizing terms and tokens to synthesize new responses (i.e., generation-based conversation), or finding existed responses which are deemed appropriate to reply (i.e., retrieval-based conversation). People work hard on how to maintain relevant, meaningful and user-engaging conversations between human and computer.

Along with the prosperity of Web 2.0, people get used to having conversations which are publicly available on the websites, such as Bulletin Board System (BBS) forums, social media (e.g., *Facebook*, *Twitter*) and community question-answering platforms (e.g., *Baidu Zhidao*, *Yahoo! Answers*). These resources provide a unique opportunity to build a massive collection of naturally occurring conversations. The big data propel a fast development of retrieval-based techniques for conversation studies in the open domain. The merit is that owing to the diversity on the Web, the system will be able to retrieve at least some results for any user inputs, and can be trained to flow naturalistic conversations by learning from large volumes of human-to-human conversations. In this paper, we focus on research in the retrieval-based human-computer conversation system in the open domain.

Over time, notable accomplishments in conversation systems have been achieved with tremendous efforts devoted. People have developed a well-defined paradigm and deploy it into most of existed conversation systems: given a human utterance as the *query*, the computer returns a *response*. The standard procedure presumes that humans take the role to lead a conversation.

An interesting thought comes to our mind. Since query suggestion has been demonstrated to be helpful for classic information retrieval, and a search-based conversation system applies the retrieval paradigm, what if we introduce a *proactive* conversation mode with "response ranking" as well as "next utterance suggestion"? Query suggestion brings information outside of a particular user's scope and provides what others might search given the entire dataset, leading to better search experiences. So can we expect the utterance suggestion could improve the conversation experiences?

We believe the answer is yes. There are several advantages for a proactive human-computer conversation mode featured with next utterance suggestion. One typical situation is that the system predicts something that the user is going to express (similarly as query suggestion might do). In this scenario, the user just needs to click the suggestion and it becomes more convenient to talk. Moreover, conversations in the open domain generally do not limit to only one ground truth response, and people keep an open mind in chit-chat. By taking next utterance suggestions, users are able to continue a conversation about new contents that they might not originally intend to talk about. Thirdly, one more critical issue is that there can be "stalemates" during a human-computer conversation [14]. A passive conversation system cannot break the stalemates but a proactive one can, by suggesting appropriate next utterances to say. Due to manifold reasons, we expect better human-computer conversation experiences in the new proactive conversation mode.

We have two questions to answer: 1) does the proposed new conversation task really work? 2) If the task works, how to formulate the problem and design a model to solve it?

We briefly formulate the new task. Given a query q, the conversation system provides a pair of response and suggestion (r,s). r is to respond q, while s is to suggest how the conversation continues given r. In contrast to the traditional passive conversation style, the system now provides additional information to attract more user engagement, which is a "proactive" conversation style. The new mode is featured with next utterance suggestion and response ranking, simultaneously.

We can decompose the task into several subtasks. The most straightforward method is to find an appropriate response given the query, and then to find a good suggestion given the response in the same way. However, we will show that the simple solution does not work. For one thing, the suggestion is not guaranteed to be relevant to the query; moreover, if a response diverges from the query, a suggestion based on the response would become meaningless.

To this end, we propose a dual recurrent neural network chains with Long-Short Term Memory (LSTM) units for the new conversation task, namely Dual-LSTM Chain Model (*Dual-LSTM*). The model formulates two matching chains for *query-response* and *response-suggestion* separately, and couples the dual chains in an end-to-end joint learning manner. The information across the words is propagated through chains, and both chains are tangled to fuse input information via a unified way, which is a novel insight.

We conduct extensive experiments in a variety of human-computer conversation setups and evaluation metrics in terms of p@1, MAP, nDCG and MRR. We run experiments against several rival algorithms to answer the two questions that we brought up previously: is the task useful and is the model effective? The results are positive.

To sum up, we have several contributions as follows:

• To the best of our knowledge, we are the first to systematically investigate the proactive conversation mode with next utterance suggestion. The joint learning problem and our task formulation are novel[1].

• To solve the joint learning problem, we propose a novel Dual-LSTM Chain Model based on the recurrent neural networks. The proposed model simultaneously matches a query with a pair of response and suggestion as the next utterance.

• We conduct extensive experiments to investigate whether the proactive conversation mode is useful, and to examine the effectiveness of the Dual-LSTM model. By studying the phenomenon of the system equipped with the new conversation mode, we have some interesting discoveries.

The rest of the paper is organized as follows. We start by reviewing related work in Section 2. We introduce the task statement in Section 3. In Section 4, we describe our proposed model. We devise experimental setups and evaluations against a variety of baselines and discuss results in Section 5. We draw conclusions in Section 6.

2 RELATED WORK

For decades, researchers work on human-computer conversation systems. In early days, people generally focus on conversation systems established by rules or templates [30, 34]. The idea is simple and such methods require no data or few data for training, while instead require a great many human efforts to create enough handcraft rules or templates to run the system. To build rule-based systems is costly. Yet a conversation goes out of scope easily. People begin to pay more attention to data-driven methods.

From human-driven conversation systems to data-driven conversation systems, the need for a much bigger amount of data is substantially increasing. Nowadays, with the prosperity of social media (such as microblogs), forums and other Web 2.0 resources, people have conversations with each other on the Internet, publicly. It is feasible to collect a very large amount of human-to-human conversation data [33].

With massive data available, it is intuitive to build a retrieval-based conversation system as information retrieval techniques are developing fast. Given a user utterance as a query, the system searches for candidate responses by certain matching metrics. Leuski *et al.* build systems to select the most suitable response to the query from the question-answer pairs using a statistical language model as cross-lingual information retrieval [8]. The database consisting of a number of question-answer pairs is a key to success [9].

[1] Google works on utterance assistance in Allo and Gmail system with a vision of "responding without typing". So basically, the technique only produces a candidate response (w/o a next utterance). Our problem formulation with *response-suggestion* pairs is quite different.

Researchers propose to augment the database with question-answer pairs retrieved from plain texts [2, 19].

Another way to build a conversation system is to use language generation techniques. Higashinaka *et al.* propose to combine language template generation with the search-based methods [3]. Ritter *et al.* have investigated the feasibility of conducting short text conversation by using statistical machine translation (SMT) techniques, learning from millions of naturally occurring conversation data in Twitter [21]. In these approaches, a response is generated from a model, not retrieved from a repository, and thus it cannot be guaranteed to be a legitimate natural language text [42].

In recent years, deep neural networks (DNNs, also known as *deep learning*) have made a significant improvement. DNNs are highly automated learning machines; they can extract underlying abstract features of data automatically by exploring multiple layers of non-linear transformation [1]. Prevailing DNNs for sentence-level modeling include convolutional neural networks (CNNs) and recurrent neural networks (RNNs). In CNNs, we have a fixed-size sliding window to capture local patterns of successive words [7], whereas RNNs keep one or a few hidden states, and collect information along the word sequence in an iterative fashion [39]. Due to a well-known problem of gradient vanishing/explosion in vanilla RNN, Long-Short Term Memory (LSTM) and Gated Recurrent Unit (GRU) are proposed to address the issue [20, 28].

With the help of deep learning techniques, generation-based conversation systems are greatly advanced. In general, the conversation system applies the sequence-to-sequence generation manner [28]. A neural responding machine is proposed for single-turn conversation [24]. Soon, the conversation is extended into multi-turns, with plain contexts [26] and hierarchical contexts [23, 29]. Researchers gradually introduce various elements into conversation generation, such as diversity [12], persona [13], topic [38], knowledge [40] and additional contents [18].

Needless to say, retrieval-based conversation systems are also greatly advanced using neural networks. A series of information retrieval-based methods are applied to short-text conversations using microblog data, either for single-turn conversation [6, 11, 15] or multi-turn conversation [42, 43, 45]. Basically, these methods model sentences using convolutional [4, 15] or recurrent [20, 31] units to construct abstractive representations. Many matching metrics are proposed for retrieval using deep neural networks. Palangi *et al.* have proposed sentence matching based on vector similarities [20]. A recursive schema is later introduced for incremental sentence modeling [31, 32]. Usually, sentences are compared in a pairwise matching style via word-by-word matchings, known as sentence pair modeling [4, 41]. The chain-based matching is also demonstrated to be useful, where the first sentence's information is available when modeling the second one [10, 22]. In this paper, we use the chain-based matching with recurrent units. Auxiliary information such as topics [37], knowledge [36], and contents [14] can also be incorporated for better retrieval. Although not all of these methods are originally designed for conversation, they are very effective for short-text matching tasks in general. We include some of them as strong baselines in the experimental setups.

In this paper, we propose a new conversation task of response ranking and next utterance suggestion. The difference compared with related work is quite clear. The mentioned studies basically rank responses by certain matching metrics. Most traditional human-computer conversations are passive: the computers need only to respond. No content will be suggested to continue the conversation. Here in the new conversation mode, the computers become proactive to suggest next utterances when deemed appropriate. We investigate the new conversation task and evaluate its usefulness in the application scenario. Secondly, we propose a Dual-LSTM Chain Model to couple the joint learning task in this paper, which is also a novel insight.

3 TASK STATEMENT

3.1 Problem Formulation

The new conversation task contains two hops of retrieval. Given a query q from the human, the computer would retrieve several candidate responses r to respond. For each r, the system will retrieve a bunch of candidate suggestions s as next utterances. The two-hop process is illustrated in Figure 1. After the two hops of retrieval, we have one query to several responses, and each response is followed by several suggestions. We formulate them as triples, each as $\{q,r,s\}$.

For each triple of $\{q, r, s\}$, we feed them into the proposed neural network framework for sentence representation learning and semantics matching. The ranking score $\mathcal{F}(.)$ is a learned function from all evidences through an end-to-end learning. The best matched r and s will be output in pairs as an appropriate response and a next utterance suggestion. In contrast to a traditional conversation formulation as $r^\star = \text{argmax}_r\ \mathcal{F}(r|q)$, we formulate the joint learning task for a proactive conversation system as follows:

$$(r, s)^\star = \underset{r,s}{\text{argmax}}\ \mathcal{F}\big((r, s)|q\big) \tag{1}$$

We decompose the joint learning task into subtasks, and then analyze the major technical issues for each of them.

Subtask 1 (Response Ranking.) *Given a user issued query* $\boldsymbol{q}$*, we retrieve several candidate responses* $\boldsymbol{r}$ *from the conversational dataset* $\mathcal{D}$*. We learn to match* $\boldsymbol{r}$ *with* $\boldsymbol{q}$*.*

It is undoubted that the relevance between the query and the response is a key issue to the joint learning task. If a selected response is not related to the query, the entire task might become meaningless. Subtask 1 aims at modeling the semantic correlation between the query and the response. Here we apply an LSTM chain to characterize the semantic relevance and to measure whether a response is appropriate.

Subtask 2 (Suggestion Ranking.) *Given a candidate response* $\boldsymbol{r}$*, we retrieve several candidate suggestions* $\boldsymbol{s}$ *from the conversation dataset* $\mathcal{D}$*. We learn to match* $\boldsymbol{s}$ *with* $\boldsymbol{r}$*.*

Another key issue for the task is the relevance between the (selected) suggestion and the (selected) response. Intuitively, if the suggestion is not related to the response, the

task would make less sense. This subtask characterizes the semantic matching between a response and a suggestion. We can apply another LSTM chain to model the relevance then decide how likely to give a suggestion as the next utterance.

Yet, these two LSTM chains should not be isolated. Since the choice of the first chain (a response) and the choice of the second chain (a suggestion) are provided as a pair, both components need to be somehow coupled together. Moreover, since part of the first chain is also part of the second chain, the first LSTM chain can have a positive impact (when appropriately matched), or negative penalty (when mismatched), on the second LSTM chain. In other words, the second LSTM chain will be influenced by an external "memory" from the first LSTM chain and the memory is a soft gating mechanism for the information propagation.

Subtask 3 (Joint Coupled Learning.) *Given the candidate responses* $\boldsymbol{r}$ *and suggestions* $\boldsymbol{s}$*, we learn to couple them together so as to rank a pair of* $(\boldsymbol{r}, \boldsymbol{s})$ *given* $\boldsymbol{q}$*.*

From subtasks 1 and 2, we can measure the correlations between a query and a response, as well as a response and a suggestion. Neither of the matching result will be output in isolation: they are further coupled through deep neural networks and we obtain a final ranking score for (r,s) pairs.

We believe that training such an end-to-end model to minimize a single objective function, and with minimum reliance on hand-crafted features, will yield superior performance in the long run. We introduce the Dual-LSTM Chain Model by taking into account of the 1) query-response relevance, 2) response-suggestion relevance, and 3) the joint learning. We will elaborate the components in Section 4.

3.2 System Pipeline

We briefly describe the system pipeline and data flow. Roughly, we have an off-line process and an online process.

Off-line Process. As mentioned, people now interactively have conversations with each other on the Internet, which provides a huge thesaurus of conversational data. Therefore we are able to collect sufficient human-to-human conversation data taken from the Web. Users post *messages* visible to the public, and then receive *replies*. Such conversations do not occur in strict real-time style but they are literally real human conversations which last for one turn or more. We collect the conversational data ($\mathcal{D}$), stored as "virtual documents" (d), each virtual document as d=($message$,$reply$) and $\mathcal{D}=\{d_i\}_{i=1}^{|\mathcal{D}|}$. The whole corpus is formatted in an inverted index prepared off-line. Besides, we train the matching model off-line to learn human conversations from the data.

Online Process. After taking in a user-issued *query*, we apply a standard retrieval process via keyword search from $\mathcal{D}$. To be more specific, we search for a *message* using the *query*, and returns the *reply* as the *response* to output, as shown in Figure 1(a). Similarly, we search for a *suggestion* given a candidate *response* using the same conversation repository $\mathcal{D}$, shown in Figure 1(b). For more details of how to retrieve responses from a conversational database, interested readers may refer to more elaborations in [42, 43]. The two-hop of retrieval returns a list of potentially relevant *responses* and *suggestions* out of the corpus given the *query*. We feed them into the learned matching model.

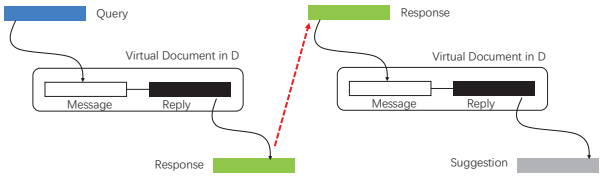

Figure 1: Two hops of retrieval to get *responses* **and** *suggestions* **from the virtual documents given a** *query*. **In general, the retrieval process is to search for the** *message* **part and to return the** *reply* **part.**

4 DUAL-LSTM CHAIN MODEL

To learn the representation and matching is the core component in the retrieval-based conversation system. We establish a joint learning framework. Two LSTM chains combine the ranking evidences and determine the match of a response-suggestion pair to a query. All ranking evidences are integrated together through the proposed Dual-LSTM Chain Model shown in Figure 2.

4.1 LSTM Chain

To be self-contained, we first give a quick overview of word embedding and the neural network structures.

4.1.1 Word Embeddings. In text-related models, a word generally acts as an atomic unit; thus, the internal relation between similar words might lose. A typical approach is to map a discrete word to a dense, low-dimensional, real-valued vector, called an *embedding* [17]. Each dimension in the vector captures some (anonymous) aspect of underlying word meanings. This process is known as vectorization. Given enough data, word embeddings can make highly accurate guesses about the meaning of a particular word. Embeddings can equivalently be viewed that a word is first represented as a one-hot vector and multiplied by a look-up table [17].

In our model, we first vectorize all words using their embeddings, which serve as the foundation of our deep neural networks. Word embeddings are initialized randomly, and then tuned during training as part of model parameters.

4.1.2 LSTM Networks. We use the recurrent neural networks (RNNs) with LSTM units to propagate information along the word sequence. RNNs keep a hidden state vector, which changes according to the input at each time step.

LSTM is an advanced type of RNN by further using memory cells and gates to learn long term dependencies within a sequence to address gradient vanishing or explosion problems [20, 28]. LSTM maintains a memory cell that updates and exposes its content only when deemed necessary. LSTM models are defined as follows: given a sequence of inputs, an

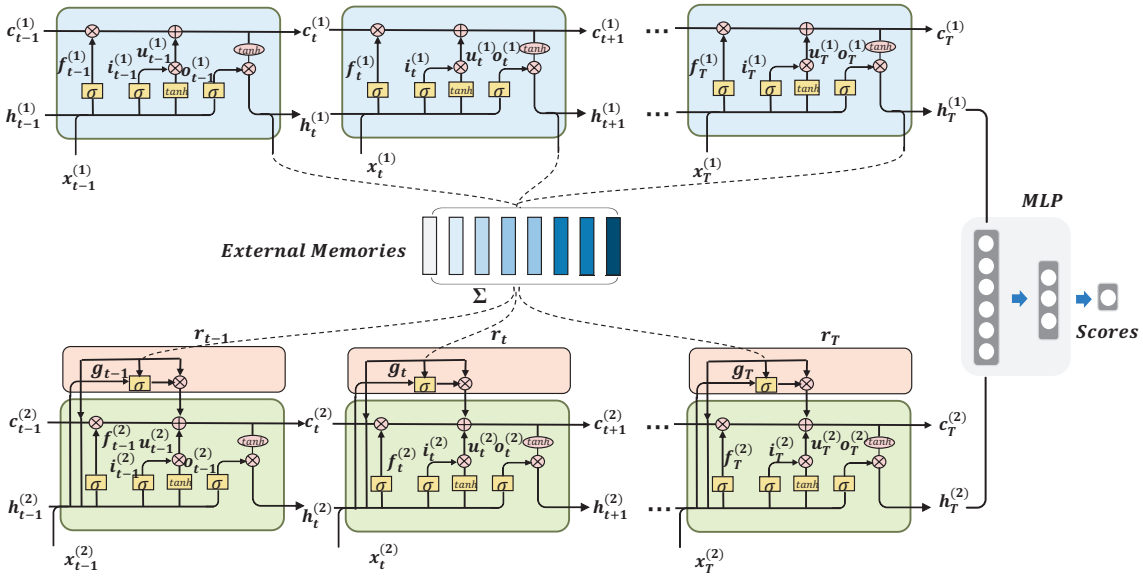

Figure 2: Dual-LSTM Chain Model with the first LSTM chain to characterize *query-response* and the second LSTM chain to characterize *response-suggestion*. The two LSTM chains are coupled by an external memory states and a multi-layer perceptron (MLP).

LSTM associates each position with *input*, *forget*, and *output gates*, denoted as $\mathbf{i}_t$, $\mathbf{f}_t$, and $\mathbf{o}_t$ respectively. The vector $\mathbf{c}_t$ is used to additively modify the memory contents. Given an input sentence $\mathbf{x} = \{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T\}$, where $\mathbf{x}_t$ is the vector of word embedding for individual token (i.e., a word) at position t in the sentence. LSTM outputs a representation $\mathbf{h}_t$ by combining $\mathbf{h}_{t-1}$ and $\mathbf{x}_t$ for position t, given by

$$\begin{bmatrix} \mathbf{i}_t^{(1)} \\ \mathbf{f}_t^{(1)} \\ \mathbf{o}_t^{(1)} \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \end{bmatrix} (W \cdot \begin{bmatrix} \mathbf{h}_{t-1}^{(1)}, \mathbf{x}_t^{(1)} \end{bmatrix} + \mathbf{b}) \tag{2}$$

$$\mathbf{u}_t^{(1)} = \tanh(W \cdot \left[\mathbf{h}_{t-1}^{(1)}, \mathbf{x}_t^{(1)}\right] + \mathbf{b}) \tag{3}$$

$$\mathbf{c}_t^{(1)} = \mathbf{f}_t^{(1)} \odot \mathbf{c}_{t-1}^{(1)} + \mathbf{i}_t^{(1)} \odot \mathbf{u}_t^{(1)} \tag{4}$$

$$\mathbf{h}_t^{(1)} = \mathbf{o}_t^{(1)} \odot \tanh(\mathbf{c}_t^{(1)}) \tag{5}$$

where $\mathbf{b}$ indicate the bias and $\odot$ denotes element-wise multiplication. $\sigma(\cdot) = \frac{1}{1+e^{-\cdot}}$ is a known as a sigmoid function. Intuitively, the forget gate controls the amount of which each unit of the memory cell is erased, the input gate controls how much each unit is updated, and the output gate controls the exposure of the memory state, which is stored in the internal memory cell.

For a pairwise matching, two sentences are matched via a transformation of an affinity matrix through parameters trained and learned from the network. In this paper, we use the chain-based matching through information propagated in the LSTM sequence. Such a method has been proved effective for matching and alignments [10, 22].

In this way, we obtain a series of state $\{\mathbf{h}_t^{(1)}\}$ with a superscript $^{(1)}$ from the first LSTM chain, and the final hidden state $\mathbf{h}_T^{(1)}$ which indicates the information from the query and response. As indicated in Section 3.1, the hidden states $\{\mathbf{h}_t^{(1)}\}$ will be utilized to characterize the other LSTM chain. We take the output vector $\mathbf{h}_T^{(1)}$ as a feature representation and feed it to a feed-forward neural work for future integration and optimization. The gradients are computed using the back-propagation algorithm [35].

4.2 Dual-LSTM Chain

We incorporate another LSTM chain for response-suggestion matching. The gist is similar to the first LSTM chain. Additionally, the first LSTM chain has an impact by propagating information through a soft gating mechanism on the second LSTM chain. We will first explain the external "memory" from the first LSTM chain. We formulate a relevance vector through a gating cell into the neural network units in the second LSTM chain so as to control information fusion.

4.2.1 External Memories. We use external memories constructed by history hidden states $\{\mathbf{h}_t^{(1)}\}$ from the first LSTM chain between the query and the response, which is:

$$\mathrm{M} = \{\mathbf{h}_1^{(1)}, \mathbf{h}_2^{(1)}, \ldots, \mathbf{h}_T^{(1)}\}$$

where $\mathbf{h}_t^{(1)}$ is the hidden state at time t emitted by the first LSTM chain. The memory blocks M are used to store the external information for matching. Hence, the history information can be read from the memory block. We denote a relevance vector (denoted as $\mathbf{r}$) from external memories as $\mathbf{r}_i$, which can be computed by soft attention mechanisms:

$$\mathbf{r}_i = \sum_1^T \alpha_{ij} \mathbf{h}_j^{(1)} \tag{6}$$

where α represents the attention distribution over the external memory states M.

The attention signal can be calculated as:

$$\alpha_{ij} = \text{Softmax}\big(\phi(\mathbf{h}_j^{(1)}, \mathbf{x}_i^{(2)}, \mathbf{h}_{i-1}^{(2)})\big) \quad (7)$$

where

$$\phi(\mathbf{h}_j^{(1)}, \mathbf{x}_i^{(2)}, \mathbf{h}_{i-1}^{(2)}) = \mathbf{v}^T \tanh(W \cdot \Big[\mathbf{h}_j^{(1)}, \mathbf{x}_i^{(2)}, \mathbf{h}_{i-1}^{(2)}\Big] + \mathbf{b}) \quad (8)$$

W are weight matrices, $\mathbf{v}$ is weight vector and $\mathbf{v}^T$ denotes its transpose. Here we use superscripts $^{(1)}$ and $^{(2)}$ to denote the vectors from the first LSTM chain and the second LSTM chain respectively. The score is based on how to control the information fusion from the query-response relevance. The attention schema is parametrized as a neural network which is jointly trained with all the other components [22, 24].

4.2.2 Gating Cell. With additional information of the external memory blocks, we have more evidence to measure the relevance between a response and a suggestion. The information from the first LSTM sequence is maintained and passed along for further judgments of the second LSTM chain. Our motivation is to propagate "reliable" relevance evidence from the first LSTM chain to measure the second one. To this end, we design a gating cell to read relevance memories into the second LSTM chain.

The gating cell is to control the information fusion into the second LSTM chain: relevance evidence can pass through the gate and irrelevant evidence shall not. In this way, the semantic relevance from the first chain implicitly performs as the clue for the selection of the second chain. Starting from the original vectors, at each time step the cell decides the alignment with the first LSTM chain, and decides what information should be retained for future time steps and discards the others. This cell plays the role of gating relevant information fusion, and the semantic clues can be integrated into the second LSTM chain smoothly.

We add a new gating cell, and the relevance information is incorporated into the LSTM units. The cells read the relevance vector, and decides whether or how to use the external memory from the first LSTM chain into the second one.

$$\begin{bmatrix} \mathbf{i}_t^{(2)} \\ \mathbf{f}_t^{(2)} \\ \mathbf{o}_t^{(2)} \\ \mathbf{g}_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \sigma \end{bmatrix} (W \cdot \begin{bmatrix} \mathbf{h}_{t-1}^{(2)}, \mathbf{x}_t^{(2)}, \mathbf{r}_t \end{bmatrix} + \mathbf{b}) \quad (9)$$

$$\mathbf{u}_t^{(2)} = \tanh(W \cdot \Big[\mathbf{h}_{t-1}^{(2)}, \mathbf{x}_t^{(2)}, \mathbf{r}_t\Big] + \mathbf{b}) \quad (10)$$

$$\mathbf{c}_t^{(2)} = \mathbf{f}_t^{(2)} \odot \mathbf{c}_{t-1}^{(2)} + \mathbf{i}_t^{(2)} \odot \mathbf{u}_t^{(2)} + \mathbf{g}_t \odot \mathbf{r}_t \quad (11)$$

$$\mathbf{h}_t^{(2)} = \mathbf{o}_t^{(2)} \odot \tanh(\mathbf{c}_t^{(2)}) \quad (12)$$

After updating the new LSTM units, the hidden state of $\mathbf{h}_T^{(2)}$ will be passed as the output of the second LSTM chain.

4.3 Coupled Chains

Now we have two final states denoted as $\mathbf{h}_T^{(1)}$ and $\mathbf{h}_T^{(2)}$ from the Dual-LSTM chains. We concatenate both vector representations together, and feed the concatenated vector to an ensuing network for further information mixing. Vector concatenation for matching is also applied in various studies like [44], which is effective yet of low complexity order.

The joint vector is then passed through a 3-layer, fully-connected, feed-forward neural network, also known as *multi-layer perceptron* (MLP) [1], which allows rich interactions. The network enables to extract features automatically, starting from lower-level representations to higher-level ones, till the system provides an overall judgment of the appropriateness for the query, response, and suggestion triple.

We have two options to formulate the task-specific output. We can formulate the whole task as a ranking problem. For ranking tasks, the outputs are scalar matching scores. We can also formulate the task as a classification problem, when the outputs are the probabilities of the different classes, which are computed by the softmax function on the matching vectors. In this paper, we use the first formulation of a ranking problem. A single neuron outputs the matching score of both components. As mentioned, $\mathcal{F}((r, s)|q)$ is in $\mathbb{R}$; the final scoring neuron is essentially a linear regression.

Here we apply hinge loss to train the proposed network. Given a triple of positive sample (q, r^+, s^+) in the training set, we randomly sample a negative instance r^- and/or s^-. The objective is to maximize the scores of positive samples while minimizing that of the negative samples. Concretely, we would like the score of positive samples $\mathcal{F}(+)$ to be as least the score of negative samples $\mathcal{F}(-)$ plus a margin Δ. Thus, the training objective is to

$$\underset{\theta}{\text{minimize}} \sum \max\{0, \Delta + \mathcal{F}(+) - \mathcal{F}(-)\} + \lambda\|\theta\|_2^2 \quad (13)$$

where we add an ℓ_2 penalty with coefficient λ for all the parameters θ which are weight and bias values optimized by the network from all ranking evidences.

As our model is (almost) everywhere differentiable, the parameters of the networks are optimized with stochastic gradient descent using the back prorogation algorithm to compute the gradients. The gradients can be propagated all the way back through coupling, matching, and individual sentence modeling. In this way, heterogeneous information can be incorporated organically with our model under the unified deep framework. According to the objective function to optimize in Equation (13), it is sufficient to learn the model by computing the gradients with respect to the model parameters. The computation for these gradients can be decomposed using chain rules.

4.4 Model Variants

Our proposed model has some natural connections to existed neural network models for conversation tasks. We add some more discussions about the model variants.

Model Degeneration. We have two LSTM chains to feed into the MLP function. If we just omit the semantic matching which comes from the second LSTM chain, the model degenerates to a traditional matching method for the standard human-computer conversation without next utterance suggestion. In other words, a model for the standard human-computer conversation is part of our proposed model.

Another degeneration is that we separate the proposed model into two isolated LSTM chains without any coupling structures. We use the first chain to match a *query* and a *response*, and use the second chain to match a *response* and a *suggestion*. The model degenerates into a Chain-LSTM model [10], which is the most intuitive way to tackle the proactive conversation mode with response ranking and next utterance suggestion. Although not coupled, two isolated chains still provide two ranking lists for either responses or suggestions. We include the model in experimental analysis.

Model Extension. There are two typical scenarios for conversations: single-turn conversation and multi-turn conversation with "context" information from a session [42, 43]. Since the proactive conversation mode with response ranking and next utterance suggestion is quite a new study, we start from the basic single-turn conversation scenario to verify its usefulness: the matching is measured among a triple of a *query*, a *response* and *suggestion*.

It is exciting to see that with small extensions, the proposed model will be compatible with the multi-turn conversation scenario. We can encode the contexts into the first LSTM chain as people generally do [23, 26]. The information from the first LSTM chain can still be propagated to the second LSTM chain, the Dual-LSTM model can be used for multi-turn conversation scenarios. We leave further investigations as our future work.

5 EXPERIMENTS AND EVALUATION

The objectives of our experiments are 1) to examine whether the new proactive mode with a next utterance suggestion is useful in human-computer conversation systems, and 2) to verify the effectiveness the Dual-LSTM model for the task.

5.1 Dataset

We use the data which contain a large number of human conversations crawled from open Web, where the users publish a message visible to the public, and then receive a bunch of subsequent replies to their utterances. We conducted data filtering and cleaning procedures by removing extremely short utterances and those of low linguistic quality.

We constructed another conversation dataset of 1,606,583 samples to train the proposed model, 357,018 for validation, and 11,097 for testing. It is important that the dataset for learning does not overlap with the data for retrieval: we strictly comply with the machine learning regime. For each training and validation sample, we randomly chose inappropriate responses and/or suggestions to obtain a negative sample. Validation was based on the model accuracy.

5.2 Hyperparameters

We use 128-dimensional word embeddings, and they were initialized randomly and learned during training. As our dataset is in Chinese, we performed standard Chinese word segmentation. We maintained a vocabulary of 177,044 phrases by choosing those with more than 2 occurrences.

The LSTM units have 128 hidden units for each dimension. We used stochastic gradient descent (with a mini-batch size of 100) for optimization, gradient computed by standard back propagation. Initial learning rate was set to 0.8, and a multiplicative learning rate decay was applied. The above parameters were chosen empirically. We used the validation set for early stopping.

During the experiments, a suggestion will be provided when the matching score of the triple exceeds a threshold (empirically set as 0.9 in this study). Otherwise, we still use the traditional conversation mode since we do not provide suggestions all the time with low confidence.

5.3 Evaluation

For evaluation, we need to examine the effectiveness of the new conversation task and the proposed Dual-LSTM model.

We need to demonstrate the proposed Dual-LSTM method is effective for the task. To this end, we evaluate the appropriateness of the response and the suggestion given a particular query. Since we can automatically construct training/testing sets, we evaluate based on 11,097 test cases. We hire annotators on a crowdsourcing platform to judge the appropriateness of top-10 ranked response and suggestion pairs from different methods for each test case. Each sample was judged by annotators via majority voting based on *appropriateness*. "1" denotes appropriateness and "0" indicates otherwise.

The evaluation design consists two parts: the evaluation interfaces are shown in Figure 3. The annotator will first be shown two sentences s_1 and s_2. They are asked whether the s_2 is good to respond s_1. In this way, we obtain the matching results for either *query-response* (recorded as Type I evaluation results) or *response-suggestion* (denoted as Type II evaluation results). We keep the appropriate pairs when s_1 is the query and s_2 is a response. Next s_3 is added into the evaluation, and annotators are asked to judge whether s_3 is appropriate given both s_1 and s_2. We know whether s_3 is a good suggestion given s_1 and s_2 (denoted as Type III evaluation). Other triples are annotated as inappropriate.

Given the ranking lists with scores, we evaluated the performance in terms of precision@1 (p@1), mean average precision (MAP) [27], and normalized discounted cumulative gain (nDCG) [5]. Since the system outputs the highest ranked results, p@1 is the precision at the 1st position, and should be the most natural way to indicate the fraction of suitable result among the top-1 triple retrieved. We also provide the top-10 ranking list for all test queries using nDCG and MAP, which test the potential for a system to provide more than one appropriate triples. We aimed at selecting as many

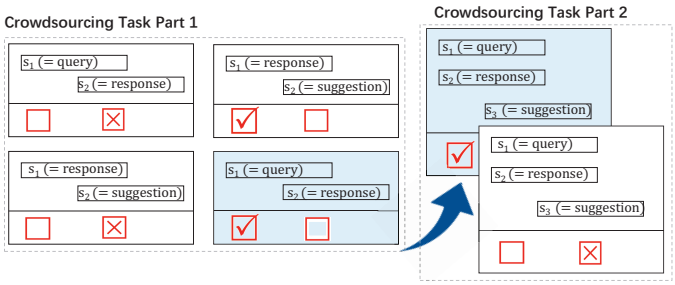

Figure 3: Interfaces for crowdsourcing evaluations. In Crowdsourcing Task 1, annotators are asked to judge whether a second sentence s_2 is appropriate to the first sentence s_1. Next we filter the results from Task 1, and keep the samples when s_1 is a test *query* and appropriateness judged as *yes* for Task 2. In Task 2, annotators judge whether the third sentence s_3 is appropriate given the matched s_1 and s_2.

appropriate candidates as possible into the top-10 list and rewarding methods that return suitable triples on the top.

Formally, the metrics are computed as follows.

$$\text{nDCG@i} = \frac{1}{|\mathcal{T}|}\sum_{q\in\mathcal{T}}\frac{1}{Z}\sum_{i=1}^{k}\frac{2^{\psi_i}-1}{\log(1+i)}$$

where $\mathcal{T}$ indicates the testing query set, k denotes the top-k position in the ranking list, and Z is a normalization factor obtained from a perfect ranking. ψ_i is the relevance score for the i-th candidate reply in the ranking list (i.e., 1: appropriate, 0: inappropriate).

MAP is computed by

$$\text{MAP} = \frac{1}{|\mathcal{T}|}\sum_{q\in\mathcal{T}}\frac{1}{N_q}\sum_{i=1}^{k}P_i\times\psi_i$$

Here N_q is the number of appropriate candidates selected, and P_i is the precision at i-th position.

Since we use real triplets collected from multi-turn human-human conversations for testing, we also have the human result added in the ranked list after matching. We include the Mean Reciprocal Rank (MRR) evaluation computed as:

$$\text{MRR} = \frac{1}{|\mathcal{T}|}\sum_{q\in\mathcal{T}}\frac{1}{rank(q)}$$

where $rank(q)$ is the position of the human result from the real triplet within the candidate ranking list.

Unlike MAP and nDCG, which examine the ranks of appropriate candidates, Mean Reciprocal Rank (MRR) focuses on evaluating the capability of retrieval systems to find (perhaps) the best result which means the positive triples created by humans in our dataset. MRR is useful but does not test the full capability because there can be multiple appropriate utterances to continue a conversation.

5.4 Competing Algorithms

We include several algorithms as baselines to compare. Since our proposed approach is technically a retrieval method, we mainly focus on retrieval-based conversation system. For fairness we use the same pre-processing procedure and the same data for all algorithms. Given a query, the systems return matched responses and suggestions accordingly.

- *Okapi BM25.* We include the standard retrieval technique to rank candidates. For each utterance, we retrieve the most relevant results using BM25 model [16].
- *ARC-II.* There are many convolutional kernel based sentence matching method [4, 15]. The ARC-II approach is a typical neural network based method with convolutionary layers which construct sentence representations and produce the final matching scores via a MLP layer [4].
- *LSTM-RNN.* A sentence is encoded as a vector representation by the last hidden state from an LSTM-RNN sequence. Two sentences are matched by cosine similarity in pairs [20].
- *MV-LSTM.* MV-LSTM is a deep architecture for word-to-word matching of two sentences with multiple positional sentence representations by LSTM sequences [31].
- *Chain-LSTM.* The Chain-LSTM indicates another matching style. When concatenated as a chain, the first sentence helps to model the second one. The information from both sentences interweaves sentence modeling and matching [10].
- *Dual-LSTM.* We propose the Dual-LSTM Chain Model based on several novel insights: two matchings are modeled by two LSTM chains, and the dual chains are coupled by joint learning through deep neural networks. We have some model variants and we discuss the analysis in Section 5.6.

5.5 Overall Performance

We conduct the appropriateness evaluation to see the performance of all methods. We have three types of results from the crowdsourcing evaluations. We measure the appropriateness between a query and a response (Type I), the appropriateness between a response and a suggestion (Type II), and most importantly, the appropriateness between a query and a (response, suggestion) pair (Type III). For baselines, they do not have the coupling component, thus we adapt two separate matchings for *query-response* and *response-suggestion*. The results are shown in three rows in Table 1.

Okapi BM25 represents the standard keyword-based retrieval method. The performance for BM25 is moderate, not surprising. In general, BM25 only utilizes the shallow representation of individual terms, while it is less capable to capture the deep semantics inside sentences.

Deep learning systems are demonstrated to have stronger capabilities to learn the abstractive representation [1, 7, 25]. The deep learning algorithm groups clearly outperform the shallow learning method. Measuring the interactions of each and every terms from the two sentences at every position can be ascribed as a classic matching style, either as a convolutional baseline *ARC-II*, or recurrent baseline *MV-LSTM*. Word-to-word matching in every position is better than the

Table 1: Appropriateness results of Type I, II, and III evaluations. Bold fonts indicate the acceptance of improvement hypothesis for the *joint learning* task (shown in gray rows) by Wilcoxon test at a significance level of 0.01. nDCG means nDCG@10.

Model	p@1	MAP	nDCG	MRR	
Okapi BM25	0.272	0.253	0.302	0.169	I
	0.259	0.226	0.284	0.156	II
	0.138	0.126	0.187	0.091	III
ARC-II	0.394	0.294	0.421	0.232	I
	0.387	0.291	0.415	0.217	II
	0.255	0.201	0.278	0.142	III
LSTM-RNN	0.338	0.283	0.371	0.228	I
	0.351	0.300	0.366	0.237	II
	0.206	0.195	0.233	0.128	III
MV-LSTM	0.435	0.322	0.409	0.308	I
	0.410	0.313	0.414	0.301	II
	0.269	0.251	0.267	0.168	III
Chain-LSTM	0.416	0.328	0.429	0.301	I
	0.422	0.316	0.410	0.307	II
	0.261	0.246	0.298	0.183	III
Dual-LSTM	0.431	0.339	0.441	0.312	I
	0.442	0.326	0.437	0.319	II
	0.425	**0.315**	**0.419**	**0.303**	III

matching from a single position (i.e., *LSTM-RNN*), which concurs the observation in [31].

The single *Chain-LSTM* method is regarded as another matching regime, and is demonstrated to be useful in other matching tasks such as entailment [22]. We have similar observations. The MRR score indicates that this method is good at capturing information flows through a conversation.

For all baselines, shallow or deep, pairwise or single-chain matching, responses and suggestions are isolated. There is no guarantee for baselines to couple both components in order to avoid conversation divergence from the query. Our proposed method performs well for Type I and II tasks, and it combines two isolated components together and shows an overwhelming improvement in Type III task. It is as expected: the information from the first match should be used to keep a response and the following suggestion in line.

5.6 Discussions

Now we have validated that our proposed model is good to tackle the task. Next we will take a closer look at the components, the structure and different model variants for more analysis and discussions.

5.6.1 Analysis. We have two LSTM chains, and we check the model with different components and variants. For the Dual-LTSM family, the first variant is to degenerate the full model into two isolated LSTM chains, which is exactly the Chain-LSTM model included in the baselines. We omit this variant due to strict page limits. We also degenerate the model by removing 1) the gating cell with external memories and 2) the coupling MLP layer, separately. We observe that performance drops compared to the full Dual-LSTM model. The results are shown in Table 2.

Table 2: Component analysis for model variants.

Model	p@1	MAP	nDCG	MRR	
-MLP	0.422	0.326	0.432	0.304	I
	0.425	0.320	0.428	0.310	II
	0.271	0.248	0.313	0.216	III
-Cell	0.426	0.333	0.438	0.308	I
	0.430	0.331	0.422	0.317	II
	0.392	0.299	0.401	0.289	III
Dual-LSTM	0.431	0.339	0.441	0.312	I
	0.442	0.326	0.437	0.319	II
	0.425	0.315	0.419	0.303	III

This phenomenon indicates both coupling designs are useful for the joint learning task. The performance drops when we remove the gating cell, which explains that cells are used as soft attention from external memories. The performance drops even more when we remove the MLP layer, since the MLP layer directly mingles information from two chains together. We also have an interesting observation that compared with the Chain-LSTM method, the model variant with an MLP layer (i.e., *-Cell*) performs better even for the response ranking subtask. We assume that seeing the information from the "future" conversations helps ranking, which is another justification for the proactive conversation mode.

6 CONCLUSION

In this paper, we propose a novel conversation mode between humans and computers, featured with both response ranking and next utterance suggestion.

We propose a joint learning model for the new conversation task. Since we have two components of responses and suggestions to rank, we formulate a Dual-LSTM Chain Model to characterize each component and couple them together with deep neural networks. We examine the effect of the proposed model against a series of baselines. Our method consistently outperforms the baselines in terms of p@1, MAP, nDCG, and MRR. We also investigate experiments to analyze different components in Dual-LSTM.

To sum up, our contributions are that 1) we propose a new conversation paradigm, 2) a novel joint learning model, and 3) to verify the usefulness of the task and the model. This work opens to several interesting directions of future studies, such as next utterance suggestions for multi-turn conversations with session contexts available. We leave the idea as the future work for further investigations.

ACKNOWLEDGMENTS

We thank the reviewers for their insightful comments. This research is partially supported by 863 Grant No. 2015AA015403,

NSFC Grant No. 61672058, and by Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology).

REFERENCES

[1] Yoshua Bengio. 2009. Learning deep architectures for AI. *Foundations and Trends in Machine Learning* 2, 1 (2009), 1–127.

[2] Gao Cong, Long Wang, Chin-Yew Lin, Young-In Song, and Yueheng Sun. 2008. Finding Question-answer Pairs from Online Forums. In *SIGIR*. 467–474.

[3] Ryuichiro Higashinaka, Kenji Imamura, Toyomi Meguro, Chiaki Miyazaki, Nozomi Kobayashi, Hiroaki Sugiyama, Toru Hirano, Toshiro Makino, and Yoshihiro Matsuo. 2014. Towards an open domain conversational system fully based on natural language processing. In *COLING*.

[4] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *NIPS*. 2042–2050.

[5] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated Gain-based Evaluation of IR Techniques. *ACM Trans. Inf. Syst.* 20, 4 (2002), 422–446.

[6] Zongcheng Ji, Zhengdong Lu, and Hang Li. 2014. An Information Retrieval Approach to Short Text Conversation. *CoRR* abs/1408.6988 (2014).

[7] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188* (2014).

[8] Anton Leuski, Ronakkumar Patel, David Traum, and Brandon Kennedy. 2009. Building effective question answering characters. In *SIGDIAL*. 18–27.

[9] Anton Leuski and David Traum. 2011. NPCEditor: Creating virtual human dialogue using information retrieval techniques. *AI Magazine* 32, 2 (2011), 42–56.

[10] Chaozhuo Li, Yu Wu, Wei Wu, Chen Xing, Zhoujun Li, and Ming Zhou. 2016. Detecting Context Dependent Messages in a Conversational Environment. In *COLING'16*. 1990–1999.

[11] Hang Li and Jun Xu. 2014. Semantic matching in search. *Foundations and Trends in Information Retrieval* 8 (2014), 89.

[12] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A Diversity-Promoting Objective Function for Neural Conversation Models. In *NAACL'16*. 110–119.

[13] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A Persona-Based Neural Conversation Model. In *ACL'16*. 994–1003.

[14] Xiang Li, Lili Mou, Rui Yan, and Ming Zhang. 2016. StalemateBreaker: A Proactive Content-Introducing Approach to Automatic Human-Computer Conversation. In *IJCAI'16*. 2845–2851.

[15] Zhengdong Lu and Hang Li. 2013. A Deep Architecture for Matching Short Texts. In *NIPS*. 1367–1375.

[16] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Vol. 1.

[17] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[18] Lili Mou, Yiping Song, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2016. Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation. In *COLING'16*. 3349–3358.

[19] Elnaz Nouri, Ron Artstein, Anton Leuski, and David R Traum. 2011. Augmenting Conversational Characters with Generated Question-Answer Pairs. In *AAAI Fall Symposium: Question Generation*.

[20] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2015. Deep Sentence Embedding Using the Long Short Term Memory Network: Analysis and Application to Information Retrieval. *arXiv preprint arXiv:1502.06922* (2015).

[21] Alan Ritter, Colin Cherry, and William B. Dolan. 2011. Data-driven Response Generation in Social Media. In *EMNLP'11*. 583–593.

[22] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2016. Reasoning about Entailment with Neural Attention. In *ICLR*.

[23] Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models. In *AAAI'16*. 3776–3783.

[24] Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural Responding Machine for Short-Text Conversation. In *ACL-IJCNLP'15*. 1577–1586.

[25] Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *EMNLP'11*. 151–161.

[26] Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A Neural Network Approach to Context-Sensitive Generation of Conversational Responses. In *NAACL'15*. 196–205.

[27] Hiroaki Sugiyama, Toyomi Meguro, Ryuichiro Higashinaka, and Yasuhiro Minami. 2013. Open-domain Utterance Generation for Conversational Dialogue Systems using Web-scale Dependency Structures. In *SIGDIAL*. 334–338.

[28] Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *NIPS'14*. 3104–3112.

[29] Zhiliang Tian, Rui Yan, Lili Mou, Yiping Song, Yansong Feng, and Dongyan Zhao. 2017. How to Make Contexts More Useful? An Empirical Study to Context-Aware Neural Conversation Models. In *ACL'17*.

[30] Marilyn A. Walker, Rebecca Passonneau, and Julie E. Boland. 2001. Quantitative and Qualitative Evaluation of Darpa Communicator Spoken Dialogue Systems. In *ACL*. 515–522.

[31] Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2016. A Deep Architecture for Semantic Matching with Multiple Positional Sentence Representations. In *AAAI'16*. 2835–2841.

[32] Shengxian Wan, Yanyan Lan, Jun Xu, Jiafeng Guo, Liang Pang, and Xueqi Cheng. 2016. Match-SRNN: Modeling the Recursive Matching Structure with Spatial RNN. In *IJCAI'16*. 2922–2928.

[33] Hao Wang, Zhengdong Lu, Hang Li, and Enhong Chen. 2013. A Dataset for Research on Short-Text Conversations.. In *EMNLP*.

[34] Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *SIGDIAL*. 404–413.

[35] Ronald J Williams and Jing Peng. 1990. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural computation* 2, 4 (1990), 490–501.

[36] Yu Wu, Wei Wu, Zhoujun Li, and Ming Zhou. 2016. Knowledge Enhanced Hybrid Neural Network for Text Matching. *arXiv preprint arXiv:1611.04684* (2016).

[37] Yu Wu, Wei Wu, Zhoujun Li, and Ming Zhou. 2016. Topic Augmented Neural Network for Short Text Conversation. *arXiv preprint arXiv:1605.00090* (2016).

[38] Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2016. Topic Augmented Neural Response Generation with a Joint Attention Mechanism. *arXiv preprint arXiv:1606.08340* (2016).

[39] Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *EMNLP'15*. 1785–1794.

[40] Zhen Xu, Bingquan Liu, Baoxun Wang, Chengjie Sun, and Xiaolong Wang. 2016. Incorporating Loose-Structured Knowledge into LSTM with Recall Gate for Conversation Modeling. *arXiv preprint arXiv:1605.05110* (2016).

[41] Rui Yan, Liang Kong, Congrui Huang, Xiaojun Wan, Xiaoming Li, and Yan Zhang. 2011. Timeline Generation Through Evolutionary Trans-temporal Summarization. In *EMNLP'11*. 433–443.

[42] Rui Yan, Yiping Song, and Hua Wu. 2016. Learning to respond with deep neural networks for retrieval-based human-computer conversation system. In *SIGIR'16*. 55–64.

[43] Rui Yan, Yiping Song, Xiangyang Zhou, and Hua Wu. 2016. Shall I Be Your Chat Companion?: Towards an Online Human-Computer Conversation System. In *CIKM'16*. 649–658.

[44] Biao Zhang, Jinsong Su, Deyi Xiong, Yaojie Lu, Hong Duan, and Junfeng Yao. 2015. Shallow Convolutional Neural Network for Implicit Discourse Relation Recognition. In *EMNLP'15*. 2230–2235.

[45] Xiangyang Zhou, Daxiang Dong, Hua Wu, Shiqi Zhao, Dianhai Yu, Hao Tian, Xuan Liu, and Rui Yan. 2016. Multi-view response selection for human-computer conversation. In *EMNLP'16*. 372–381.